Proposing mechanisms to increase the reliability between distributed devices in the IEC61499 Standard

Pranay Jhunjhunwala

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 24.5.2021

Supervisor

Prof. Valeriy Vyatkin

Advisor

Dr Udayanto Dwi Atmojo

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Pranay Jhunjhunwala

Title Proposing mechanisms to increase the reliability between distributed devices in the IEC61499 Standard

Degree programme Electrical and Automation Engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC0007

Supervisor Prof. Valeriy Vyatkin

Advisor Dr Udayanto Dwi Atmojo

Date 24.5.2021 **Number of pages** 53 **Language** English

Abstract

The need for smarter and efficient production in the Industry 4.0 era is being achieved with the help of component-based architecture. Control applications designed using the IEC61499 standard and based on the component-based architecture can be distributed over various devices connected to one-another via the wired or the wireless medium. While the distribution of control application on different devices drastically increases the flexibility and modularity, the reliability across the control application can be affected due to the increased needs of communication between the devices. In this thesis, an advanced handshake message verification algorithm has been developed and tested, to ensure the reliability between devices communicating over a lossy wireless channel. The mechanism has been designed so as to be easily integrated in the existing control application and perform the reliability operations over the existing communication network.

Keywords IEC 61499, Reliability, Handshaking, Communication

Aalto University
School of Electrical Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Pranay Jhunjhunwala

Työn nimi Ehdotetaan mekanismeja hajautettujen laitteiden välisen luotettavuuden lisäämiseksi IEC61499-standardissa

Koulutusohjelma Elektroniikka ja sähkötekniikka

Pääaine Ohjaus, robotiikka ja autonomiset järjestelmät **Pääaineen koodi** ELEC0007

Työn valvoja Prof. Valeriy Vyatkin

Työn ohjaaja Dr Udayanto Dwi Atmojo

Päivämäärä 24.5.2021 **Sivumäärä** 53 **Kieli** Englanti

Tiivistelmä

Tarve älykkäämmälle ja tehokkaammalle tuotannolle Industry 4.0 -kaudella saavutetaan komponenttipohjaisen arkkitehtuurin avulla. Ohjaussovellukset, jotka on suunniteltu käyttämällä IEC61499-standardia ja jotka perustuvat komponenttipohjaiseen arkkitehtuuriin, voidaan jakaa useille laitteille, jotka on kytketty toisiinsa langallisen tai langattoman tietovälineen kautta. Vaikka ohjaussovellusten jakelu eri laitteilla lisää huomattavasti joustavuutta ja modulaarisuutta, laitteiden välisen viestinnän lisääntyneet tarpeet voivat vaikuttaa ohjaussovelluksen luotettavuuteen. Tässä opinnäytetyössä on kehitetty ja testattu edistyksellinen kättelyviestien vahvistusalgoritmi, jolla varmistetaan häviöllisen langattoman kanavan kautta kommunikoivien laitteiden välinen luotettavuus. Mekanismi on suunniteltu siten, että se voidaan helposti integroida olemassa olevaan ohjaussovellukseen ja suorittaa luotettavuusoperaatiot olemassa olevassa tietoliikenneverkossa.

Avainsanat IEC 61499, Luotettavuus, kättely, viestintä

Aalto University
School of Electrical
Engineering

Aalto-universitetet, PB 11000, 00076 AALTO
www.aalto.fi
Sammandrag av diplomarbetet

Författare Pranay Jhunjhunwala

Titel Föreslår mekanismer för att öka tillförlitligheten mellan distribuerade enheter i IEC61499-standarden

Utbildningsprogram Electronik och electroteknik

Huvudämne Kontroll, robotik och autonoma system **Huvudämnets kod** ELEC0007

Övervakare Prof. Valeriy Vyatkin

Handledare Dr Udayanto Dwi Atmojo

Datum 24.5.2021 **Sidantal** 53 **Språk** Engelska

Sammandrag

Behovet av smartare och effektivare produktion i Industry 4.0-eran uppnås med hjälp av komponentbaserad arkitektur. Styrapplikationer utformade med IEC61499-standarden och baserade på den komponentbaserade arkitekturen kan distribueras över olika enheter som är anslutna till varandra via det trådbundna eller det trådlösa mediet. Medan distributionen av kontrollapplikation på olika enheter drastiskt ökar flexibiliteten och modulariteten kan tillförlitligheten över styrapplikationen påverkas på grund av de ökade kommunikationsbehoven mellan enheterna. I denna avhandling har en avancerad algoritm för verifiering av handskakningsmeddelanden utvecklats och testats för att säkerställa tillförlitligheten mellan enheter som kommunicerar över en förlorad trådlös kanal. Mekanismen har utformats så att den enkelt kan integreras i den befintliga styrapplikationen och utföra tillförlitlighetsoperationer över det befintliga kommunikationsnätet.

Nyckelord IEC 61499, Pålitlighet, handskakning, kommunikation

Preface

I dedicate this thesis and more over my master's degree to my maternal grandparents and my great-grand-mother, who always wished for me to study and perform well in an International Program out of India. Their blessing and guidance till the last minute has helped me become who I am today. They have scolded me, taught me and blessed me for all the years I can recall and I will continue this path to make them proud and grow day by day as a human.

I would also like to thank my parents who have always supported me, never stopped me from doing what I wanted and have always provided me with the highest level of love, care and affection. Their teaching and blessings is why because I am successful in my academic and research career.

My masters degree and this thesis would have not been possible without the guidance and the mentor-ship given to me by Professor Valeriy Vyatkin. Working under him since the early days of my masters education, he has guided me, taught me and most importantly supported me learning process all through out. I would like to offer my sincere gratitude to him for being the most supportive and helpful mentor I have had all these years. I feel privileged to be working under his guidance and mentor-ship.

Lastly, I would like to thank my friends and colleagues for always being there and helping me with any problems I've encountered during the process of my masters and thesis.

Otaniemi, 24.5.2021

Pranay

Contents

Symbols and abbreviations

Abbreviations

CFB	Composite Function Block
ECC	Execution Control Chart
EnAS	Energy Autarkic Actuators and Sensors
FB	Function Block
HLL	Higher Level Languages
HMI	Human Machine Interface
PLC	Programmable Logic Controller
SOA	Service-Oriented-Architecture
SIFB	Service Interface Function Block
SIL	Safety Integration Level
SM	State Machine

1 Introduction

Industry 4.0 has brought the need for flexibility in production scenarios in the automation industry; furthermore, this growth in the demand for flexible production has simultaneously highlighted the importance of distributed and flexible automation. Distributed automation production scenarios replace large and costly controllers by various small controllers connected to one another over wireless networks. However, the need for distributed architecture and the requirement for flexibility, has revealed a gap in higher modularity standards in the industry.

A key factor in achieving higher modularity standards and inter-operability across the factory floor is the enabling of cross-vendor product integration, which can be defined as a seamless integration of devices produced and developed by different vendors. Providing such cross-vendor support compatibility is not only crucial at the physical level, but also at the level of automation architecture. Component design architecture is important at the software level to facilitate these modularity needs at the automation architecture level. Component design can be referred as programming each part of the system as individual components or a set of components, which encapsulate the implementation of the automation program, can be easily replaced, deployed, and provide a set of interfaces for easy integration with other modules of the architecture.

The IEC61499 standard[1], which is an extension to the IEC61131-3 [2] standard for programmable logic controllers (PLC), is a component-based architecture providing the necessary means for automation system developers to work and develop applications which require modularity and flexibility. IEC61499 has well defined interfaces which enable better component interactions. The standard also supports a visual component design approach which in comparison to a purely textual programming language appeals to automation systems developers. Various components and modules being connected using connection links makes the graphical programming method more attractive for developers. Even though the graphical approach makes development for automation programs easier, inter-component connections and interactions can sometimes be challenging because of the complicated and large interfaces of various modules.

Defined in the IEC61499 standard [1], adapter links, which are an integral component of the IEC61499 standard, are an efficient solution to abstract out the complexity of inter-component relationships. Adapter links combine the event and data lines within the function blocks, and are used to simplify connections and communications between various modules of the automation program, thus facilitating the integration and replacement of components as well as improving their feasibility. IEC61499 is not only a component architecture, but also a distributed architecture. This means that components may be distributed across devices and may to have communicate via networks. Even though, the IEC 61499 architecture supports the distributed deployment of control applications to various different controller which help with the flexibility and modularity, it comes at a cost of reduced reliability because of wireless communication of the distributed devices.

Reliability of such connections may need to be ensured at the application level

following the end-to-end principle, which can require complex protocols. Due to the need of these end-to-end principle protocols to ensure reliability in automation solutions, developers and researchers in [3] and in [4] introduced various protocols to ensure reliability across distributed devices. However, they came at a cost of increased complexity at the application layer. These additional developments caused increased points of failure, moreover, debugging at the factory-floor became complex.

This thesis aims at addressing the issue by proposing an advanced handshake mechanism to improve the reliability of wirelessly communicating devices and prototyping it with an enhanced adapter design-pattern to ensure ease of system assembly and integration. The developments for the proposal will be prototyped using the NXT Control software. The handshake message verification system will be developed as standalone Function Blocks(FB), which will then be encapsulated into a single sub-application FB and be supplemented with the advanced adapter links to ensure smooth integration into existing systems.

Validation of the developed proposal would be carried out by testing the developments using industrial production demonstrators which are operated using multiple controllers which communicate with one-another wirelessly. Between each communicating controller, we will insert these prototyped handshake message verification algorithms and then calculate the loss in messages prior and post the addition of the verification system.

The rest of this thesis is structured as follows: Chapter II covers the detailed literature review on reliability and various aspects of reliability in different domains. Chapter III highlights the IEC 61499 standard and its components, followed by the test case, the EnAS Demonstrator in Chapter IV. Chapter V explains in detail the proposed handshake message verification system, followed by the prototyping in Chapter VI. In Chapter VII a monitor to verify the operation of the handshake mechanism has been explained. Chapter VIII, explains the testing scenario used to test the developments, followed by the Results and Conclusions in Chapter IX and summary in Chapter X.

2 Literature Review

Reliability in any given sector of engineering is of prime importance. With the growth in demand for technical and engineered product, the demand for higher reliable products has seen a significant rise. Reliability according to O'Connor and Kleyner in [5] can be defined as:

"The probability that an item will perform a required function without failure under stated conditions for a stated period of time."

To achieve higher reliability standards, O'Connor defines a set of goals for reliability engineering in order of priority which are highlighted as follows:

1. Prevention of Failure : Application of prior engineering knowledge and skills to prevent the failure or reduce the probability of failure
2. Identification of Failure : Application of methods to recognize and rectify the causes of failure
3. Managing Failure : Methodologies to cope with failure when the cause has not been identified or solved
4. Methods of Estimation : Using techniques to estimate the reliability of new designs

In a general aspect, the reliability of a product, or a control application, or a software, or a communication medium, or a system is heavily dependant on the factors considered during the design process or the decisions made during the same. A cross-disciplinary design process to ensure reliability in the final product has been depicted in Figure 1.

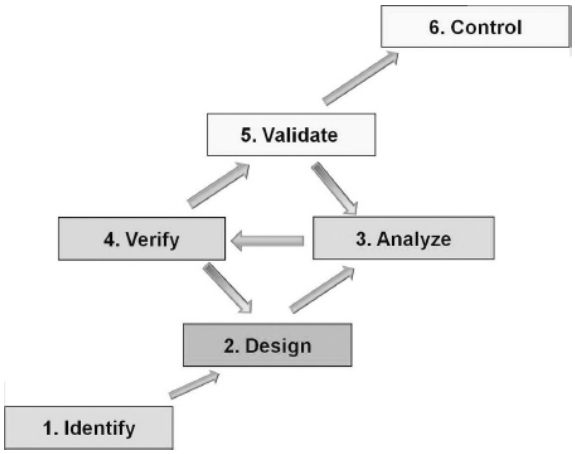

Figure 1: Activities flow for reliable design process [5].

One of the major contributors to defining guidelines for the functional safety and reliability of electrical, electronic or programmable electronic (E/E/PE), the IEC

61508 standard [6], defines four different safety integration levels (SIL), level 1 to 4, based on which systems can be analysed and verified, to determine the level of safety standards they meet or achieve. SIL based on the standard is defined as:

"A quantitative measure that indicates the degree of reliability a system must achieve to reduce the risk of failure or accident."

SIL certification to any device or system is granted based on the performance of the system with respect to the evaluation criteria set by the IEC 61508 standard. A system or device, must fulfil the following three requirements to achieve either of the SIL levels:

- System Capability : Level of reliability of product design or measure of design quality (both for hardware and software)
- Architecture Constraints : Minimum levels of safety redundancy used which may influence the reliability of the device/application
- PFH : Probability of sudden dangerous failures

The IEC 61508 standard defines generic safety rules, which have been adopted by various other industrial standards such as IEC 61131-6 in Programmable Controllers, IEC 61511 Process Industry, IEC 612061 Machinery and various other standards. In the upcoming sub-sections reliability in systems engineering, software engineering, communication, flexible and IEC 61499 have been explored.

2.1 Reliability in Systems Engineering

According the Wasson in [7] Systems Engineering can be defined as *"The multidisciplinary application of analytical, mathematical, and scientific principles to formulating, selecting, developing, and maturing a solution that has acceptable risk, satisfies User operational need(s), and minimizes development and life cycle costs while balancing Stakeholder interests."*

Reliability in systems engineering is comprised of two parts: 1) Mission Reliability 2) Equipment Reliability. The two parts are interdependent and the systems engineer has to take into account the reliability of both aspects to achieve an overall system reliability.

Mission reliability is defined in [7] as : The conditional probability that the personnel, equipment, procedural data, mission resources, system responses and facilities which comprise to form the mission system will accomplish its assigned task for the specified duration without failure.

Equipment reliability on the other hand is defined as : The conditional probability that the hardware and software, which form the equipment element of the system would deliver their individual or combined capabilities without any failure in the operating as per desired, for the desired or specified duration of time.

A failure in the system compromising its reliability can be prevented or avoided by taking into consideration the following during the system design phase:

- Design Defects : Having proper defined specifications and requirements which will help reduce or avoid design flaws, drift, incompatibility, errors
- Component Defects : Consideration of component life and degradation while system design itself
- Manufacturing Defects : Maintaining proper quality control(QC) during production and following the desired industry standards for QC
- Operational Defects : Defining the proper operating procedure, to prevent misapplication or misuse of the system
- Maintenance Defects : Defining timed maintenance and routine inspections to avoid end-of-life failures for products
- Anomaly : While system engineering and design, taking into account unexpected errors

2.2 Reliability in Software Engineering

Pham in [8], defines Software Reliability as *"The probability that software will not fail for a specified period of time under specified conditions."*

Faults in software's can cause failures which are classified into four different aspects 1) Catastrophic 2) Critical 3) Major 4) Minor. The dependence on software has increased need for reliable software's and software engineering exponentially. Even though software failures have been classified into four different categories based on the severity or the importance of the software, even a minor failure in the software can cause a major problem due to the systems and software's being linked to one another.

Software failures or errors can occur due to 1) Specification Errors 2) Software System Design or 3) Software Code Generation and these can be prevented to achieve reliability in software engineering. O'Connor in [5] highlights the following methods which can be used to achieve higher reliability standards in software engineering :

- Structured Programming : Encouraging the use of well-defined to software design, rather than using free approaches which might results in clever programs, but will be very complex and difficult to understand
- Modularity & Re-Use : Developing the code in small modules or packages which can be re-used in the application. According to the ISO/IEC90003 [9] standard, modularity reduces time and increases efficiency of code generation and fault identification
- Fault Tolerance : Development of codes for the program to automatically detect error conditions and prevent total failures by switching or going into a safety net or a safety code

- Redundancy : Numerous controllers can contain the same program or code, and a master controller can switch between the redundant devices to ensure reliability in case of controller failure
- Languages : The use of different languages can determine the reliability of systems
  - Machine Code & Assembly Level Programming : Aimed for architecture and operating systems directly because of the complexity in programming and verification
  - High Level Languages : The use of HLL's is preferred because they are processor-independent and rather use compilers to make it compatible with operating systems. Compilers once developed are extremely reliable thereby making the use of HLL's more reliable for software engineering

2.3 Reliability in Wireless Communication

In [10], reliability in wireless communication is defined as *" Wireless communication is considered reliable when a single infinitely long sequence of data packets which must be sent from the source to the destination without losses or duplicates "*

In previous works such as [11, 12, 13] it has been highlighted that there are two major researched mechanisms to ensure reliability in wireless communication between two nodes. 1) Re-transmission or 2) Redundancy

Mahmood *et al.*in [11] compares the two traditional methods of ensuring reliability and also proposes a hybrid method to use both the methods reliability and redundancy.

Re-transmission is the traditional method of improving reliability, in which the sender after sending the packet, waits for the acknowledgement of its sent packet from the receiver. However,in case the sender is not notified of any acknowledgements from the receiver, it assumes the sent packet was missed or lost during transmission. Hence, to ensure reliability, the lost packet needs to be re-transmitted.

The redundancy approach to ensure reliability, on the other instead of re-sending the full packet, the approach targets to resend only the lost or corrupted bits within the message. The sender, adds some additional information to the packet that the receiver can use to restructure the packet if some bits are lost or corrupted. The additional information includes redundant set of fragments that are encoded by employing a certain form of coding mechanism.

The hybrid mechanism proposed in [11], combines the re-transmission and redundancy method to solve the drawbacks from both. The authors highlight, in the redundancy method when the system has to undergo reconstruction of the message at the receivers end, the reconstruction will only be successful in the case when the sent fragments are equal or more than the original set of packets. In the case when the sent packets are less, the packet will be considered lost. In which case, in the hybrid mode of operation, the message will be re-transmitted using re-transmission approach.

2.4 Reliability in Distributed Flexible Automation using IEC61499

IEC 61499 standard [1], the component-based architecture in the automation industry was the first to introduce 1) distributed systems to automation 2) event driven automation 3) flexible automation.

With the rise of the standard and gradual introduction into the industry, many researchers in have identified various reliability and safety features missing in the IEC 61499 standard. Different aspects such as 1) reliability of execution 2) reliability of connection 3) proving and assessment of reliability, have been analysed in previous works carried out by researchers over these past years.

Frey and Hussian in [14], discuss three open problems 1) execution model 2) event-handling 3) data-handling, of the IEC 61499 standard which leads to unreliable verification of distributed control applications developed using the same. In [15], authors state that when a distributed control application is developed using the IEC 61499 standard and is executed using two different runtime's compliant with the standard, the results of the execution are different thereby comprising the reliability of the execution.

In [16] the authors highlight the missing granularity in the formal models used to verify the reliability and correctness of distributed control systems developed using the IEC 61499 standard. Lapp *et al.*propose the improvement of reliability by using Net Condition/Event Systems(NCES) as a formal model. The reliability of system execution, according to Lapp *et al.*can be improved by including more plant properties, such as, time-delay between the communication event in the real world, within the closed-loop verification of the models.

To improve the reliability authors in [14] have proposed 2 different ideas, a) assumption of ambiguities prior to formal verification b) improving the formal model and including additional models to compensate for the three identifies open problems.

In the work [15], a new execution runtime is proposed in which formal model verification has been introduced. The tool provides the benefits of experimenting with multiple execution models along with formal verification, thereby, ensuring the reliability of execution by proving a more stable runtime.

Garcia *et al.*in [17], introduce the combination of the CPPS architecture and IEC 61499 standard by using OPC-UA as a communication base between them. In the new architecture, reliability of the software system is guaranteed by of the use of software components with mature control algorithms. Reliability in software reconfiguration is achieved by the development of Information Components(IC) as function blocks which will be applicable for different function installations.

Authors in [18], highlight the lack of functional safety issues in the existing design patterns for the IEC61499 standard. In this work, Bhatti *et al.*propose a model-based approach to verify and analyse the reliability of the hardware and software based on both quantitative and qualitative analysis of the IEC61499 designs because according to the authors, even though standards like the IEC 61508 define safety and reliability levels and parameters individually for hardware and software, they are not sufficient for systems in which the dependence of operational reliability lies both in the hands of the software and hardware.

While, researchers have suggested these solutions, the presented works require the use of complex formal modelling methodologies to verify the reliability of the distributed control applications. These presented new runtime's involving of formal verification or conversion of function blocks into FSM require specific knowledge with is not present at the factory floor.

Researches have proposed various other prospects such as 1) development of standard design patterns 2) use of additional function blocks to enhance reliability 3) use of additional devices 4) communication protocols, to improve the identified aspects of safety and reliability.

In [3], the authors aim to improve reliability by proposing standard design patterns for the development of distributed control automation using the IEC 61499 standard. Authors in this work have identified repeatable problems and have solved it based on existing methods used in the software engineering domain.

In [19], Atmojo *et al.*highlight the lack of dependable and reliable communication within the IEC 61499 standard, for which the authors on the application level introduce standard function blocks called as channels which are inspired from the SystemJ formal programming language, and guarantee reliable communication between distributed devices. Even though researchers tackle the reliability drawback at application level in this work, there is a significant increase in the need for processing power and what is most important there is a lack of asymmetrical communication capabilities in the demonstrated work.

Zoitl *et al.*in [20] state the use of the CIP Industry protocol of communication using the EtherNet/IP as the medium of communication. Even though, CIP is an industry level accepted communication protocol that ensures reliability for frequently transfer small amounts of data, the protocol is processing intensive and uses a lot of additional parameters such as timers for the prioritization of messages in a CAN like structure. In [3] and [4], the authors have developed a basic handshake message verification with the use of adapters from the IEC 61499 standard to ensure easy system assembly and integration.

In the analysis of literature for the various reliability methods for communication, systems engineering, software engineering, distributed automation we have seen, solutions to handle reliability have been developed at the network layer, or involve the modification of hardware of to use less lossy networks or researchers have deployed additional redundant devices with the same copy of the control application to solve problems with device operation. Based on the studied literature a gap in improving reliability in existing distributed applications was identified.

In this thesis, a solution is aimed to improve the reliability between distributed controllers, communicating with one-another over a lossy wireless communication medium (2.4 GHz WiFi in the case study). The proposed development in the thesis, improves the reliability at the application layer itself and does not require any additional hardware modifications. As will be explained ahead in the the forthcoming sections, the reliability has been achieved based on a re-transmission mechanism, in which messages are sent across with special message-ID's and re-transmitted with updated ID's upon failure of communication. The developments are carried out considering the reliability methods explained in the software engineering domain, to

ensure modularity, re-use of the developed software components.

3 IEC61499 Standard and it's advantages

IEC61499[1], is a component-based architecture, which is an extension to the IEC61131-3[2], standard for programmable logic controllers(PLC). The existing IEC61131-3 is enhanced by the means of capabilities to have distributed systems and architecture.

The basic element of the IEC61499 standard is a FB and a group of FB's connected to one-another have been shown in Figure 3 and later in Figures 10 and 15 the FB interface definition can be seen. FB's in the IEC61499 standard can be of three various types 1) basic 2) composite or 3) service-interface. IEC61499 being a component-based architecture, the FB's have exhaustive interfaces that include event inputs and outputs, along with linked input and output variables.

Basic FB's in the IEC61499 are the base of the control applications developed using the standard. Basic FB not only contain the detailed interfaces of a standard FB but also support internal variables. Internal variables declared within the basic FB are secure and cannot be modified from outside because they are not displayed on the interface of the Basic FB. They can only be modified during internal processing. Execution Control Chart(ECC) shown in Figure 2 governs the processing of a Basic FB.

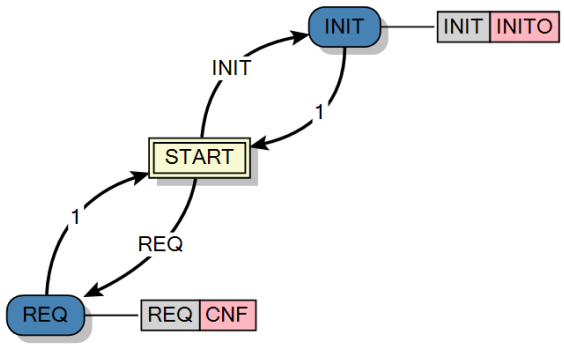

Figure 2: Execution control chart(ECC)

ECC's in the IEC61499 standard are identical to the Moore-Type state machine. ECC's have multiple states connected to each another with the help of transitions with guard conditions. In the case when the guard condition is met, the ECC would transition from one state to another. Each state in the ECC can contain either single or multiple actions, or have no action linked to the respective state. Actions comprise of two parts, an algorithm and an output event to be generated. Usage of the actions and the inclusion of algorithms or output events are all dependent on the requirements, a state could use both algorithm and event output or use either of them or use none.

IEC61499 operates on an event-driven scenario and event inputs are used to activate the FB's. The received event inputs and associated data(if any) are processed based on the ECC contained, following which event outputs can be generated based on the same ECC. FB's irrespective of their types can be connected to each other using event and data connections resulting in a Function Block Network showcased

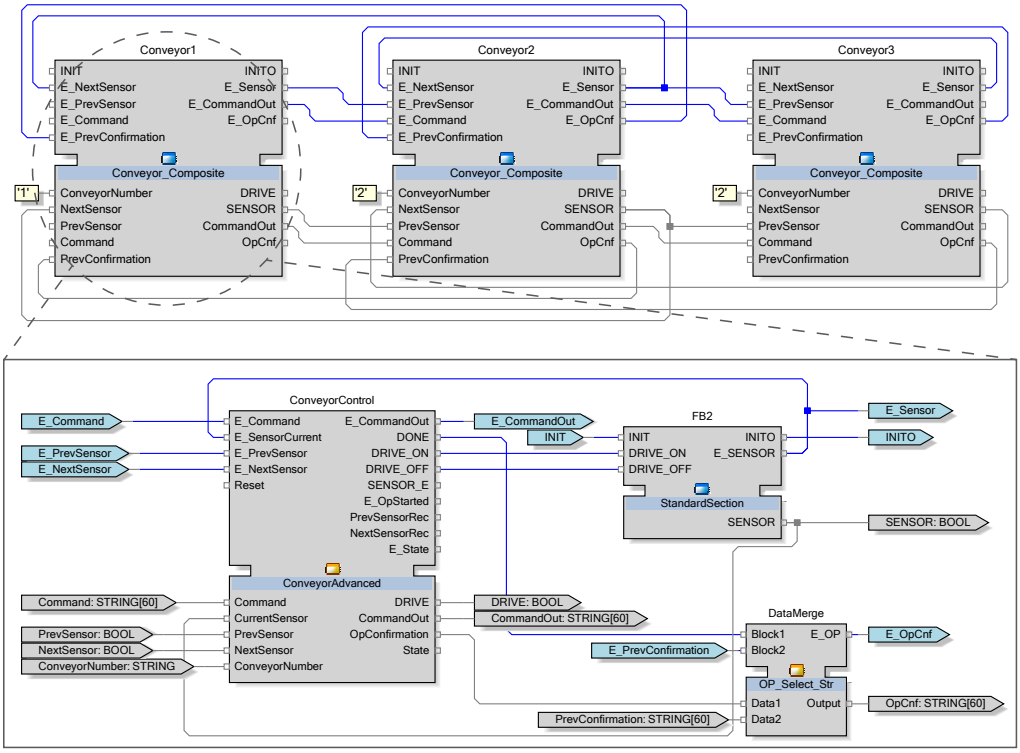

Figure 3: Composite Function Block Network and Internal Composition.

later in Figure 7. The operation of the FB network can be determined by following the event connections from one FB to another.

Composite Function Blocks(CFB), in the standard are used to combine a FB network into one large FB. A combination of CFB's and Basic FB's or network of only basic FB's together form the composition of a CFB. A major benefit of using CFB's is the possibility of developing larger hierarchical automation programs and applications. Demonstrated in Figure 3, is the connection of three CFB's along with the internal composition showcased. As can be seen in the figure, the CFB is composed of basic FB's and a CFB. Events received are processed by the blocks inside the composition and output is generated which is then transmitted to the block upstream or downstream. Once a CFB is mapped or deployed to a device, all the blocks present inside the CFB are automatically mapped to that particular devices.

IEC61499 standard was designed to enable the distributed deployment of FB's across multiple devices which has been further explained and demonstrated later in section 4, in which the control program for the application has been distributed across 9 different controllers.

IEC61499 standard also includes a different type of CFB which is known as the Sub-Application FB, with the only difference that the sub-application FB allows the deployment of their internal compositions to distributed devices. More flexibility to the application and developers is provided using the sub-application FB. In this work, we demonstrate the proposed idea using the Sub-Application technology which

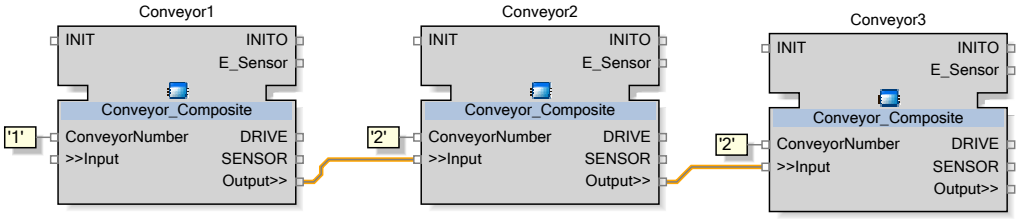

Figure 4: CFB's connected using Adapters.

has been discussed in detail in section 6.

The standard also defines an adapter technology, which further enhances the interfaces and interactions between various FB's in the network. As shown in Figure 5, adapters were introduced to make replace numerous event and data connections between various CFB's or Sub-Applications in the network by a single thick connection which would encapsulate both the events and data connections. Adapters not only encapsulate the connections but also enable two-way communication between the FB's they connect.

The 'Adapter Plug' and 'Adapter Socket' compose the adapter definition. As shown in Figure 5, plugs and sockets mirror each other's interface and plugs are defined at the output of a FB whereas sockets are defined as the input of a FB. In Figure 5, we have implemented the adapter technology above the regular data and event connections between CFB's shown in Figure 3. As we can observe event and data lines running across in both directions are encapsulated in the thick central orange connection making the network easier to access, debug and operate.

For more information on IEC 61499, we direct the reader to the proper introductory material, such as the book [21].

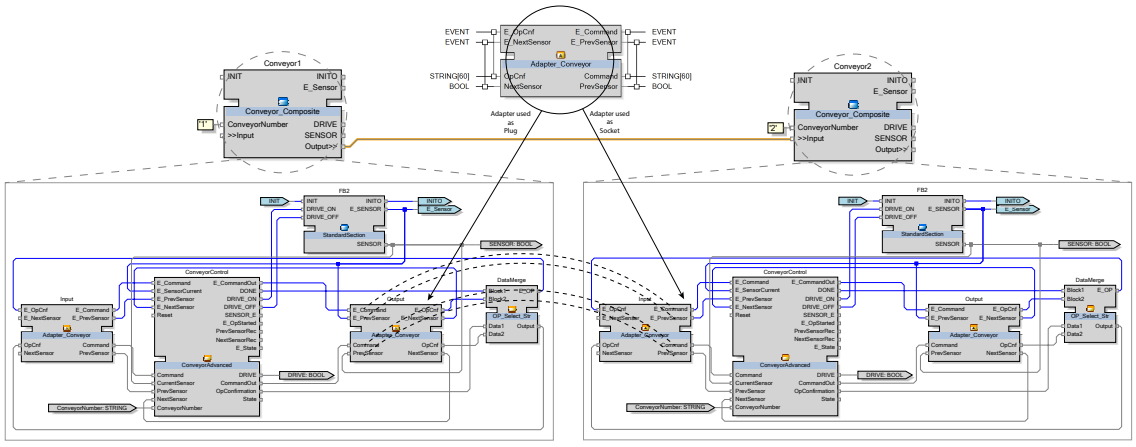

Figure 5: Adapter Interface of IEC61499.

4 Test Bed - EnAS and it's control application

Energy Autarkic Actuators and Sensors[1] commonly called as EnAS, is one of the many test beds available at the Aalto Factory of the Future[2]. Shown in Figure 6, is the EnAS production station representing a small scale manufacturing unit. The setup as can be seen consists of various mechanical components such as sets of motor driven conveyors connected in a closed-loop cyclic manner along with laser sensors on each conveyors to detect the presence of work-pieces and their motion. Also equipped with 2 sets of pneumatic operation islands consisting of jacks, sledges and grippers, EnAS provides researches with a platform to extensively test and develop various industrial automation programs, sequences and developments. The pneumatic operators are used to pick objects from the sledges and place them on the work-pieces carried by the conveyor and also remove objects being carried on the conveyor and place them on the sledge. Each sledge can hold up to two objects and moves right and left to position itself under the jacks. Grippers on the other hand are used to lift the work-piece of the conveyor making the conveyor available for other work-pieces are production requirements.

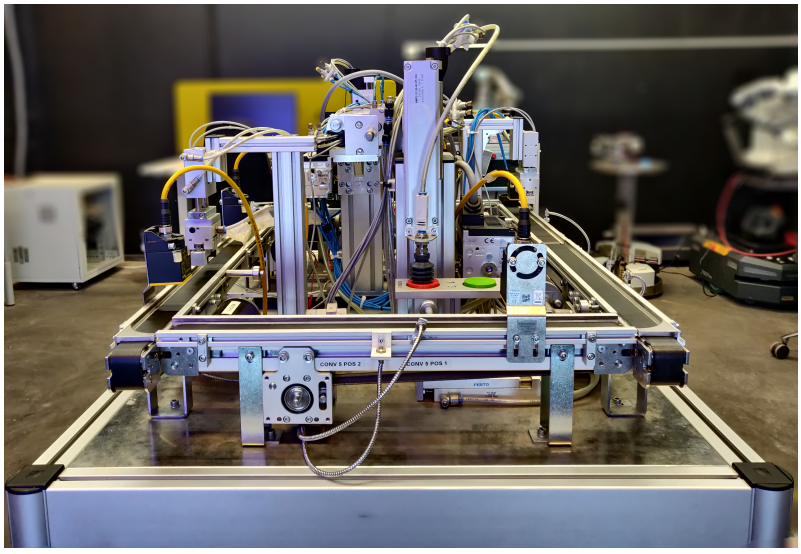

Figure 6: EnAS at the Aalto Factory of Future.

EnAS is equipped with 9 PLC's called as Ice-Blocks developed by Flexbridge[3], which operate on the IEC61499 standard and provide the means for distributed control architecture. Shown in Figure 7 is the control application developed for EnAS using the NXTStudio software by NXTControl[4] along with the distribution for the 9

[1] https://www.energieautark.com/

[2] Aalto Factory of the Future is a Research facility owned and used by the "Information Technologies in Industrial Automation" research group of Aalto University. Equipped with FESTO production platforms, Mobile AGV's, Universal Robot 3, ABB Yumi and VR facilities the lab provides researchers with various platforms to work and develop automation advancements. More information can be found at: https://www.aalto.fi/en/futurefactory

[3] https://flexbridge.se/iceblock/

[4] https://www.nxtcontrol.com/en/engineering/

controllers and also highlights the annotated 2D top-view of the demonstrator.

As highlighted in the diagram, each controller is responsible for controlling a single hardware component i.e. each conveyor has it's own controller and each pneumatic station has its own controller. There is an additional controller i.e. the 9th controller which houses the additional blocks housing the HMI, production sequences and planning services blocks such as the delivery and placement services.

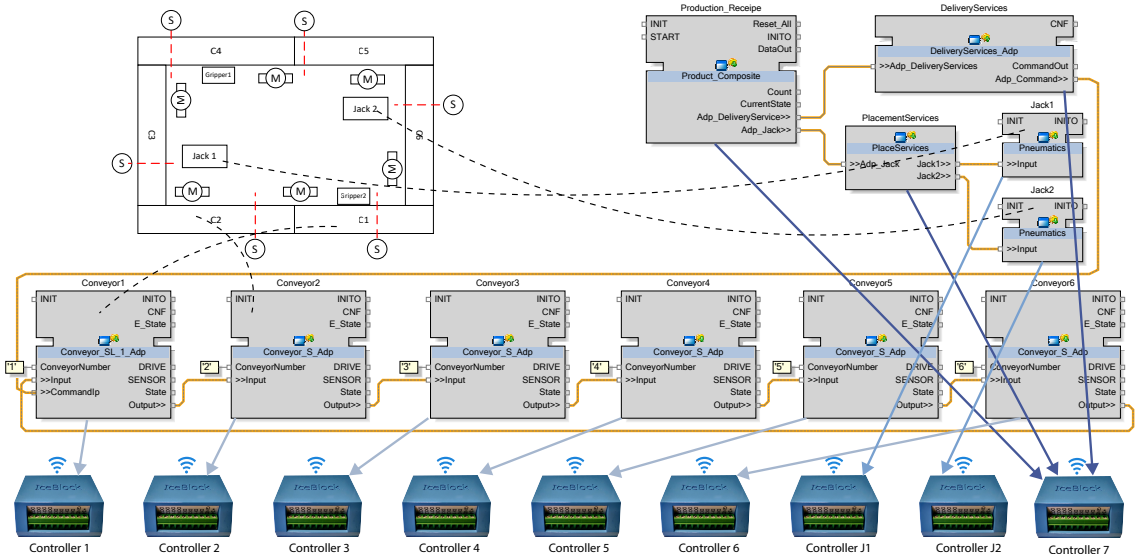

Figure 7: EnAS Control Application Deployment.

The control program for the EnAS demonstrator has been developed based on the SOA design paradigm using which control programs are developed in a layered approach in which the top level layers can reuse the services offered by the lower level layers or the layers underneath it. Figure 8, shows the control application of EnAS structured in the layered format of the SOA. The lowest layer i.e. the "Execution Services Layer" includes the services offered by each mechatronic component present in EnAS i.e. 6 conveyors along with their respective sensors and the 2 pneumatic production islands. Each of these blocks are designed as individual agents and connected in a sequential pattern based on the physical structure of the machine.

Each conveyor agent is connected using adapters links downstream and upstream to the previous and next conveyor respectively. For eg: Conveyor agent 2 is connected to conveyor agent 1 downstream and conveyor agent 3 upstream. Design of these agents is such that when commands are received from the higher level layers in the SOA based control application, each agent acts as a stand-alone unit and relies on the information from the agents downstream and upstream, thereby bringing down the communication channel occupancy by reducing the constant need of communication to the higher level planning services layer. This development pattern can be referred as the "Multi-Agent Architecture".

The second or the middle layer i.e. the "Planning Services Layer" performs the role of the bridge between the top most layer and the bottom most layer in the used architecture. This layer organises the utilization of the services in the bottom

most layer based on the requests from the top most layer i.e. the "Production Services Layer". The production services are responsible for the production scenario or sequences used to produce the products or called as "Production Recipes" as well.

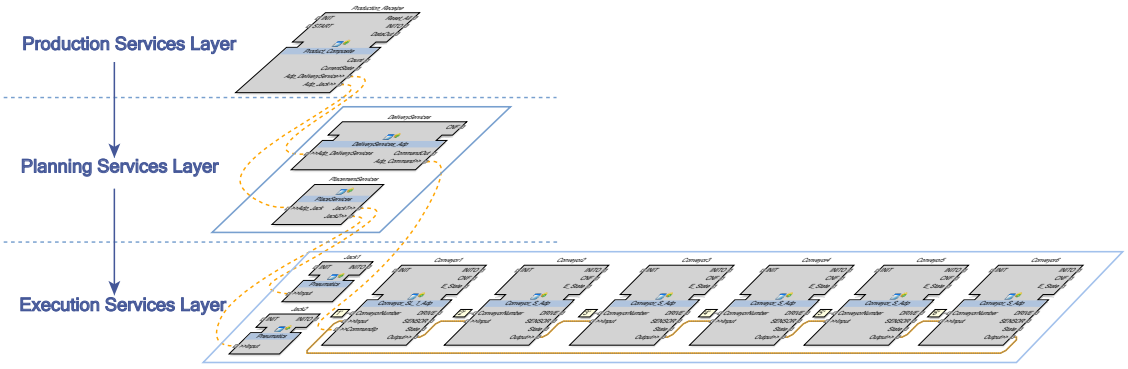

Figure 8: Service oriented architecture breakdown.

The production recipe generates request which are processed by the planning services and then the final operational commands are sent to the execution services where the mechatronic components perform the operation. For eg: in this case the production recipe would want the work-piece to go from conveyor 3 to conveyor 5, hence it would send this request to the delivery services FB in the planning services layer, which will process the request and then instruct conveyor agents 3,4 and 5 to perform the desired actions. As soon as the commands are received conveyor 3 would initiate motion and as soon as the work-piece crosses the conveyor 3 sensor, the information will be sent upstream to conveyor 4, which will start moving. Once, the work-piece crosses the conveyor 4 sensor, the sensor information will be sent upstream and downstream i.e. to conveyor agent 5 and 3 respectively. Upon reception of sensor information from conveyor 4, conveyor 3 would stop moving because it is now assured that the work-piece has crossed conveyor 4 and conveyor 5 would start moving expecting work-piece to arrive. Similar operation would happen when the work-piece reached the conveyor 5 sensor.

This sequential operation of conveyors has been designed to reduce energy consumption by running the conveyors only when they are needed. The main benefit of the SOA is the re-usability and easy of system integration. To have multiple production scenarios on the same device, automation developers need to just re-program or re-use instances of the production blocks in the top-most layer i.e. the production services and can re-use the services of the layers underneath.

The control commands for the conveyors and pneumatic islands produced by the Production_Recipe FB and the planning services block use the data Type `STRING`. Operational confirmations by the low-level agents such as conveyors and jacks are communicated upstream to the planning and production services to indicate the successful completion of the requested process. These operation confirmation messages also use the `STRING` data Type. The low level conveyor agents use the `BOOLEAN` data type to convey the sensor information to agents upstream and downstream.

The 9 controllers used to control EnAS and the production scenario communicate

over the 2.4GHz WiFi technology. Wireless communication between industrial devices over a standard communication protocol has indicated frequent packet and information loss between the PLC's causing important messages to get lost and halting the production or the operation of the device. Since various agents are dependent on controllers upstream and downstream, and also dependent on controllers responsible for the upper layers of the control program, it is vital for the communication between each controller to be reliable and efficient.

Since, the modification of the communication infrastructure i.e. the WiFi network and it's capacity was not possible due to the restrictions of the WiFi standard itself, a "Handshake Message Verification System" has been proposed in this work, which ensures successful and reliable communication between two or more wirelessly communicating devices, by counteracting the issue of the packet and information loss with the help of advanced SMs. As highlighted above communication between these devices took place using data either in the form of `STRING` type or `BOOLEAN` type, or both. Hence the proposed handshake mechanism which has been explained the upcoming section 5.

5 Handshaking and Reliability Across Distributed Devices

The need for reliability in communication across distributed devices has been highlighted in the previous section. To ensure reliability using existing automation methods and functionalities, an advanced handshake message verification mechanism was developed and proposed. The developed mechanism can be deployed across any data type, but in this work, data types `STRING` and `BOOLEAN` were used. The idea behind this handshake mechanism is to sit between any two or more FB's communicating over different devices and ensuring messages from one device reach the other.

The developed handshake mechanism has two parts which are the sender and the receiver respectively. Since the idea of this handshake is to ensure communication between two FB a pair of sender-receiver FB's is needed to ensure the reliability in one direction of communication. Various cases in which communication could have been missed and results in a message being lost were identified and have been explained in the following subsections along with the help of sequence diagram shown in Figure 9. To demonstrate the aspect of communication and working of the proposed mechanism, we take as example communication between a controller(PLC) and a motor.

5.1 Case 1: Desired Operation

Illustrated in the first section of Figure 9, is the desired operation of the handshake message verification system working between the controller and the motor. Upon reception of the message from the controller, the handshake sender appends a message ID and further transmits the message downstream to the handshake receiver. The role of the message-ID is to indicate to the receiver if the command is a new command or if its a re-transmission of the previous command. Once the message is received by the handshake receiver it is passed through a message verification algorithm which isolates the message and message-ID from one another. The message is then forwarded downstream to the respective agent(Motor in this case) and the receiver also replies back to the sender with a message confirming that it received and processed the message. To indicate the confirmation, the receiver adds an additional part i.e. ";R", to the message string which is sent to the handshake sender.

Based on the design the confirmation should be sent within a fixed time-out period which the sender initiates as soon as it sends the message to the receiver. If within this time-out period a confirmation from the receiver is not received, the handshake mechanism assumes that there has been a problem in communication and would take the desired actions based on the current state of the system.

5.2 Case 2: Sender's Message is lost during transmission

Section 2 in Figure 9 illustrates the case when a message from the sender can get lost due to a lossy or a busy network channel. As soon as the message from the

controller is received, the handshake sender forwards it with message-ID '1' and also initiates a time-out period within which it expects a confirmation for the message sent. As highlighted in the figure, the message from the sender is not delivered to the receiver, hence it does not send back a confirmation simply because according to the receiver there was no message to confirm.

Once the time-out period elapses, since the confirmation was not received, the handshake sender assumes there has been a communication error and re-transmits the same command, but with an incremented message-ID i.e. '2'. The incremented message-ID indicates that the command has been sent for the second time and is a re-transmission. In this particular case. the re-transmission would keep on continuing until the sender doesn't receive a confirmation from the receiver.

The message when received by the receiver, would pass through the message

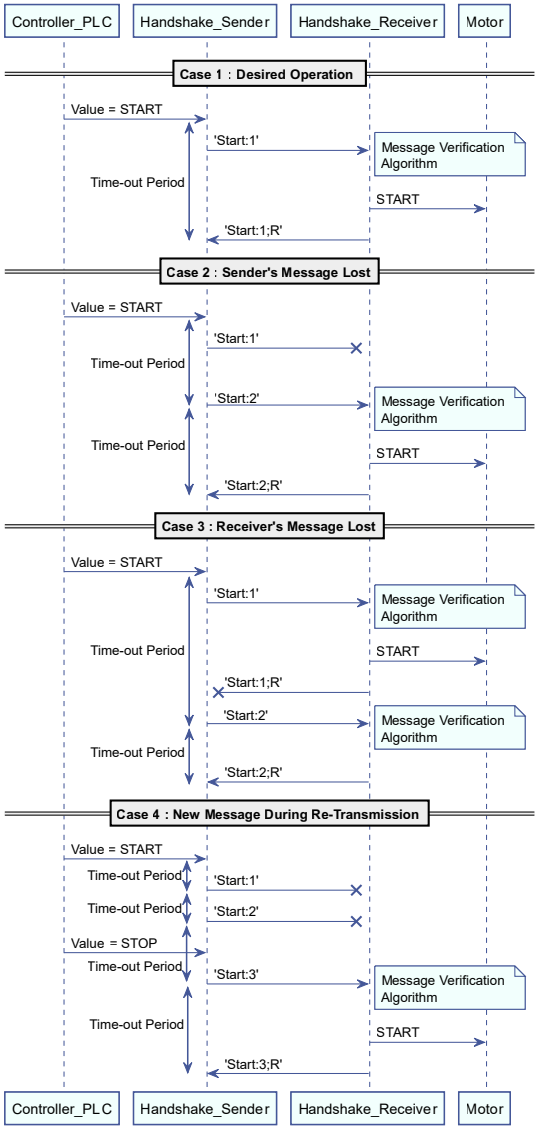

Figure 9: Handshake Message Verification Operation Chart.

verification algorithm which would check the message, message-ID and previously received commands. Since this is a new command, the command will be forwarded downstream to the motor and the receiver would send back a confirmation for the re-transmitted message i.e. 'Start:2;R'.

5.3 Case 3: Receiver's Confirmation is lost during transmission

In section 3 of Figure 9, we highlight response of the handshake message verification system in the case when a confirmation message from the receiver is lost during transmission.

In the figure we that the command 'START' was received by the sender, forwarded to the receiver and the motor was given the command to START, but the confirmation message 'Start:1;R' sent by the handshake receiver to the sender was lost in between and not delivered to the sender, because of which the sender re-transmitted the message with an updated message-ID as soon as the time-out period for the original message elapsed.

This re-transmitted message should be processed appropriately because the original command was already sent downstream to the motor and repetition of the same command could lead to larger system failures or breakdowns if not accounted for, hence, as soon as the message is received by the receiver, it is passed through the message verification algorithm. The algorithm is designed in a pattern in which each received command is thoroughly crossed checked before the receiver takes any action upstream or downstream.

In the algorithm, initially the message and the messages-ID are isolated from one-another and if the message-ID is '1' i.e. it is a new message, the command is forwarded to the agent downstream. But if the message-ID is greater than '1' it means the command has been re-transmitted in which case the verification algorithm compared the command to the previously received commands. If the previous command and the received command are same, it understands that the previous confirmation message was lost and this is a re-transmission by the sender. The verification algorithm will be further discussed in detail in the upcoming sections.

Hence as shown in case 3 in the Figure 9, when the command 'Start:2' is received, the message verification algorithm processes it and sends back a confirmation 'Start:2;R' to the sender but does not send the command downstream to the motor avoiding the case of double commands being sent.

5.4 Case 4: New Message During Re-Transmission

The most important case identified in the proposal and development of the handshake message verification system has been showcased in case 4 in Figure 9. The case highlights the situation in which the handshake sender received a new command from the controller during the time it is transmitting the previous command. As can be seen, the handshake sender is re-transmitting the START command and receives

the STOP command even before it receives confirmation for the START command message.

To handle the highlighted case, the handshake sender was advanced to account for the new incoming messages and also keeping a check on the number of re-transmissions. The handshake sender block was developed and tested in stages and the final state machine was deduced based on lab testings, which included the handshake sender blocking checking the updated or new value after a set of N re-transmissions and also checking the updated or new command after it received a confirmation for the old message. These state machines and developments have been further discussed below in the sections representing the handshake sender FB.

5.5 Handshake Mechanism Sender

The interface for the handshake mechanisms sender FB has been shown in in Figure 10. The role of this FB is to take as input the command to be sent via the input variable 'CurrentState' linked to the event 'Data_In', attach the respective message-ID and send it across to the receiver using the output variable 'DO_Adp1' linked to output event 'E_DO_Adp1'.

Along with the event output for the data, the handshake sender also controls the delay services using the 'E_Delay1' event output and received delay confirmation at the 'Delay1Done' event input. Unlike IEC 61131-3, the IEC 61499 does not support inbuilt timers inside the basic FB rather we need to call for the SIFB and use the 'E_Delay' standard FB to provide a time-delay. The time-delay is started by the sender block based on the requirements of the handshake mechanism as described earlier. The handshake mechanism sender block also additional input event and variable, 'E_DI_Adp1' and 'DI_Adp1' respectively which are used to receive the confirmation message from the receiver.

Case 4 of the proposed handshake mechanism has been tackled at the senders end i.e. with the help of the SM inside the handshake mechanisms sender block. Various SM's developed have been explained sequentially in the upcoming sections, in which we develop and test various methods to tackle the case 4 of the handshake mechanism

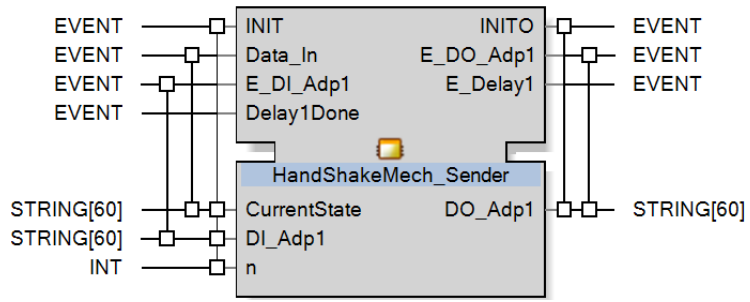

Figure 10: Handshake Message Sender Function Block.

5.5.1 Check updated-value after each transmission

The first solution proposed to resolve the issue of receiving a new message during re-transmission was to check the status of the incoming value after each re-transmission. Figure 11 shows the SM which is used to implement the proposed idea of checking after each re-transmission of the old message.

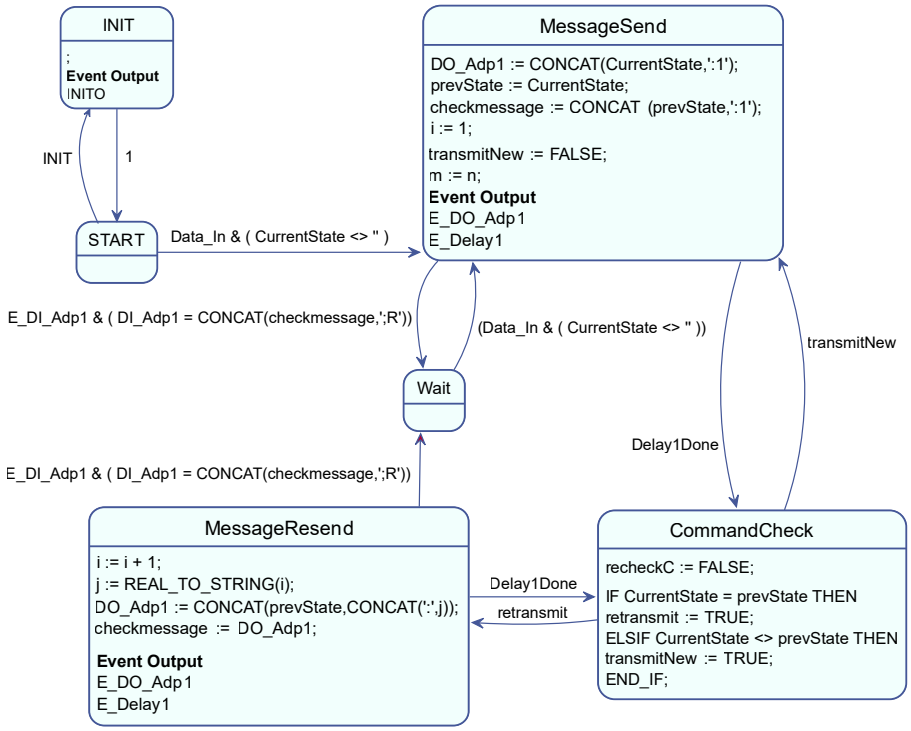

Figure 11: Handshake Mechanism Sender - Case 1.

As soon as the SM, receives the first message it transitions from state 'START' to state 'MessageSend'. In the 'MessageSend' state, the algorithm processes the new command, adds the message ID and then transmits it across to the receiver. If the sender blocks receives the message confirmation from the receiver before the time-out period, it jumps to the 'WAIT' state and waits for the new message to come or else if the time-out period elapses, it shifts to state 'CommandCheck'. For the purpose of explanation of the case, let us assume the confirmation has not been received and the sender SM shifts to state 'CommandCheck'.

When the time-out period elapses, instead of instantly re-transmitting the old message with an updated message-ID, the SM in state 'CommandCheck' first checks the input variable 'CurrentState' and compares it to the old command. If the command is same and no new command is received, the SM then jumps to state 'MessageResend' in which it re-transmits the old message but with an updated message-ID. On the contrary, if there has been a change in the command, i.e. a new command has been received during the re-transmission or the time-out period, instead of re-transmitting the old message, the SM jumps to state 'MessageSend' from state 'CommandCheck' and then the new message is transmitted.

In the case when the system is in state 'MessageResend' and the time-out period elapses, instead of just looping back into the same state, the SM jumps to state 'CommandCheck' and the process is repeated. Hence, in this SM every time the sender has to send a message, it first checks for the updates in the command and then accordingly sends the message. If there is a new command, it sends the new command with ID = 1, if the command remains the same and a re-transmission is needed, it increments the message-ID by 1 and then transmits the message.

5.5.2 Check updated-value after N transmissions

Figure 12 showcases the second SM proposed to tackle the situation presented in case 4 of the handshake mechanism. In this SM we propose to re-transmit a command 'N' time before checking the status of updated commands i.e. if the sender block is re-transmitting a message and a new command is received during retransmission of the old command, it will be only considered when the old command is re-transmitted 'N' times.

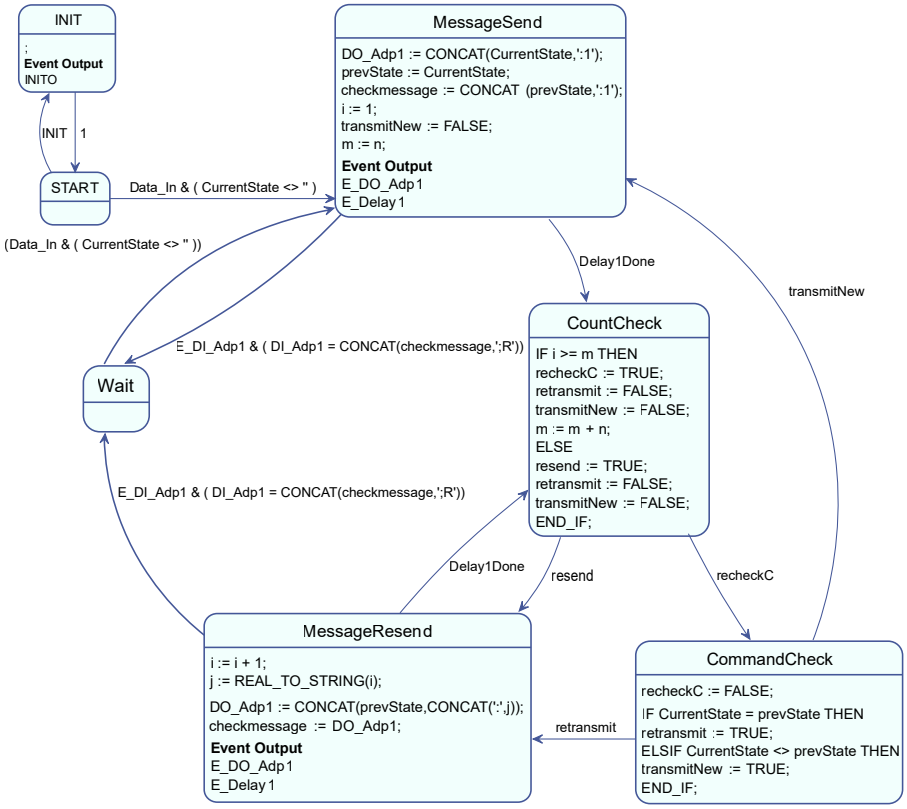

Figure 12: Handshake Mechanism Sender - Case 2

As soon as the SM, receives the first message it transitions from state 'START' to state 'MessageSend' where it transmits the message to the receiver with the addition of the message-ID to the command and also initiates a time-out period. In this case we again assume the confirmation is not received within the time-out period, following which the SM shown in Figure 12 jumps to state 'CountCheck'.

The 'CountCheck' state is the additional state in this proposal as compared to the SM in Figure 11. As soon as the time-out period elapses, before re-transmitting the old message or before checking for the new command, the SM compares the current message-ID to the input variable 'n' which defines the number of permitted re-transmissions.

If the message-ID is lesser than 'n' i.e the old command has yet not been transmitted 'n' times, the 'CountCheck' enables the internal `BOOLEAN` 'resend' which causes the SM to jump to state 'MessageResend', this processes would continue until the sender FB receives a confirmation from the receiver or in the 'CountCheck' state detects that the message-ID is equal to 'n' in which case it would set the `BOOLEAN` variable 'recheckC' TRUE. Once that happens, the SM would shift to state 'CommandCheck' in which it would check for an update in the command.

In the case when there is a new command, the system would transition to state 'MessageSend' and the process would re-initiate, but in the case when the command has not been updated, the system would instantly go back to re-transmission of the old message and would again check for 'n' re-transmissions before checking the updated state.

5.5.3 Sender State Machine

Based on the testing and analysis of the two SM's proposed above i.e. checking the updated value after each transmission and checking the updated value after 'n' re-transmissions, the need for 'n' re-transmissions was identified, but at the same time, the need for re-checking of the messages was also concluded. Hence shown in Figure 13, is the final SM that has been used in the handshake mechanisms sender FB shown in Figure 10.

While testing and analysing, it was highlighted that commands need to be re-transmitted a certain number of times to ensure accuracy and reliability during production, but the need for checking of new commands during retransmission was also seen, because in cases if an old command has been processed by the sender still keep re-transmitting the older command while the new command has already arrived, this would basically cause the system to fail or delay the production. Hence, while testing of the SM shown in Figure 12, we achieved higher levels of accuracy as compared to the SM in Figure 11 but was still very low, therefore while analysing the situation, it was highlighted that the SM in Figure 12 would re-check for the updated command at every n'th re-transmission, but if an updated command was received between transmission 'n' and 'n+n', and while the re-transmission was ongoing, a confirmation from the receiver is received before it hits the 'n+n' mark, the SM in Figure 12 would jump to state 'WAIT' and miss the new command.

The SM shown in Figure 13 is an advancement of the SM proposed in Figure 12, in which the 'WAIT' state has been also modified to check the status of the command whenever the SM enters the 'WAIT' state i.e. every time the sender FB receives a confirmation from the receiver, it would enter the 'WAIT' state in which it would then check for an updated command, if there is an updated command present, it would instantly jump to the state 'MessageSend' whereas if the command hasn't

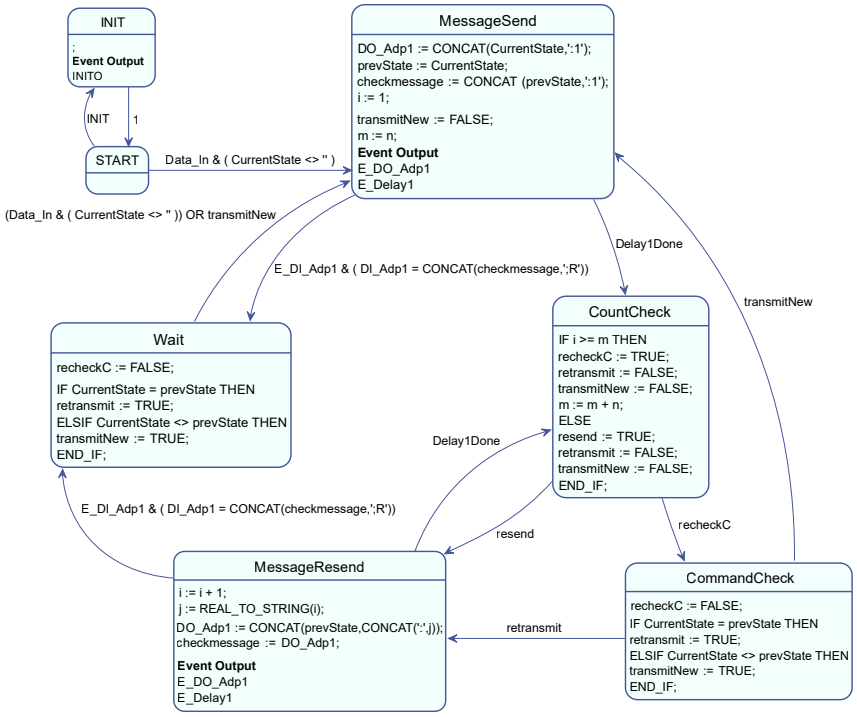

Figure 13: Handshake Mechanism Sender State Machine.

been updated, it would stay in the 'WAIT' state and wait for a command to be transmitted.

In this SM, if there is an ongoing re-transmission and during that a new command is received, the SM as desired would shift to the 'WAIT' state, but in this SM, it would again perform a 'CommandCheck' in the 'WAIT' state, thereby preventing the missing of a new command received during a re-transmission set.

Figure 14 represents the operation of the final SM shown in Figure 13. In this sequence diagram operation we assume that the value of 'n' i.e. the number of re-transmissions before checking is 5, i.e. the system would re-transmit 5 times before checking for the updated command. As highlighted in the operation, during the second re-transmission of the START command, a new command 'STOP 'from the controller is received. Once the confirmation from receiver is attained, the SM again does a Command Check and verifies that a new command was indeed received during previous re-transmissions and hence transmits the new message ensuring the smooth flow of the production scenario by maintaining constant communication reliability.

5.6 Handshake Mechanism Receiver

The receiver FB for the handshake mechanism has been shown in Figure 15. The receiver FB plays a crucial role in handling the Case 2 and Case 3 of the handshake mechanism represented in Figure 9 and explained above in sections 5.2 and 5.3. It receives as input the message from the sender using the input event and variable, 'E_DataIP' and 'DataInput' respectively.

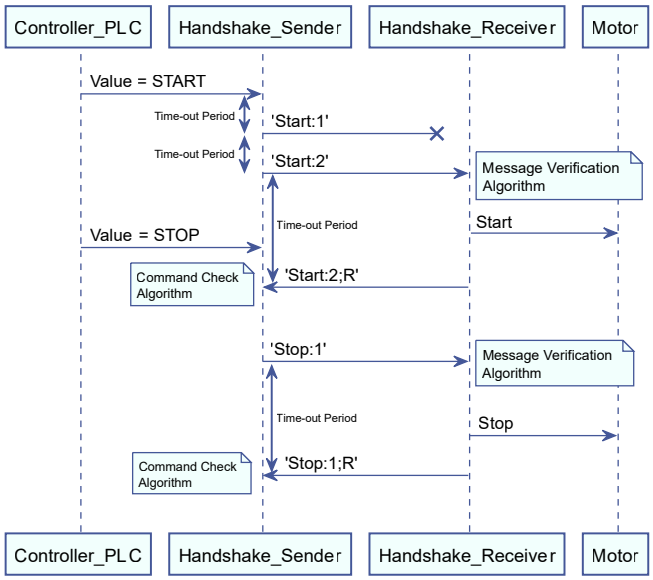

Figure 14: Handshake mechanism sender operation.

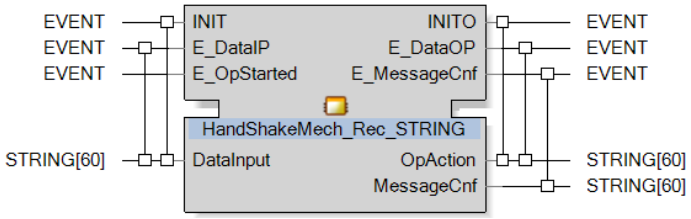

Figure 15: Handshake mechanism receiver interface.

The receiver FB houses the SM shown in Figure 16. The basic operation of the receiver FB is to isolate the message-ID from the command, and forward the command downstream using the event output 'E_DataOp' and output variable 'OpAction'. The format for variable OpAction is flexible and can be modified based on the format of command being transported(in our case or `BOOLEAN`). The receiver FB is also responsible for sending the confirmation back to the sender which is done via the 'E_MessageCnf' event output and 'MessageCnf' output variable.

The main state in this SM is the 'MessageVerification', in which it first separates the message-ID from the command, following which it analyzes the message-ID to identify if the message is a re-transmission or a new message. In the case when message-ID is equal to 1, it proceeds by sending the command downstream, followed by sending a confirmation to sender.

If the message-ID is greater than 1, then the intelligent message verification algorithm checks the command and compares it to previously processed commands and categories it into two different cases from the predefined handshake messages sequences.

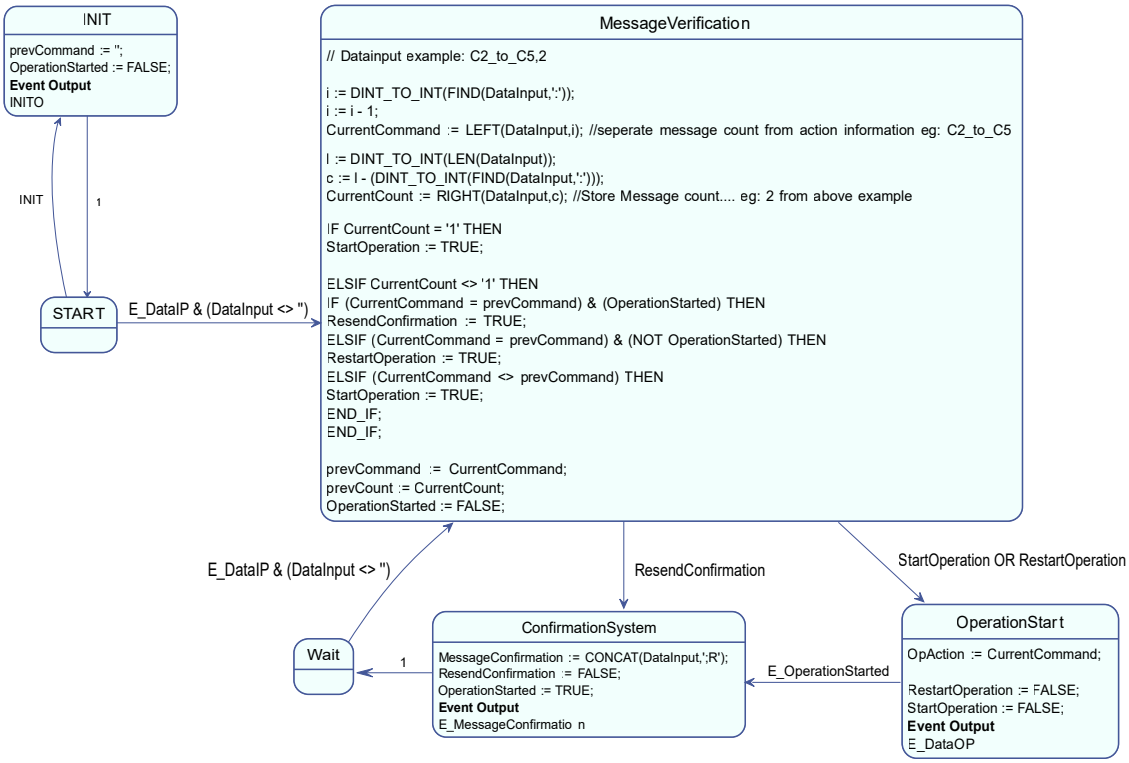

Figure 16: Handshake mechanism receiver - State machine.

5.6.1 Case 2 - New Command

In the situation, when the message is a re-transmission meaning the messaged-ID is greater than 1 and the 'MessageVerification' algorithm determines, that the command is new and differs from the previously processed command. The algorithm raises a `BOOLEAN` flag called as 'RestartOperation', which transitions the SM to the 'OperationStart' state where the command separated in the Verification algorithm is forwarded downstream. This process handles the case 2 of Figure 9 and ensures that if the first message sent by the sender was lost, the re-transmission should be processed accordingly and the process will not be hampered.

5.6.2 Case 3 - Old Command

When there is a re-transmission and the verification algorithm identifies, that the re-transmission is of a old message, in other words message-ID is greater than 1, but the command is same as the previous command, the receiver understands that the confirmation it sent to the sender was lost, and hence the sender has re-transmitted the message.

In the case, the receiver's verification algorithm will raise a `BOOLEAN` flag called as 'ResendConfirmation' indicating that only the confirmation has to be resent and the downstream block does not have to be updated as demonstrated in case 3 in Figure 9. When the 'ResendConfirmation' flag is set TRUE, the SM proceeds to state 'ConfirmationSystem', thereby skipping the 'OperationStart' phase and directly

sending a confirmation to the sender and avoiding a conflict downstream.

6 Sub Application Prototyping

To ensure ease of system assembly and simple integration of the developed handshake mechanism into existing applications, the handshake mechanism blocks were combined into a single sub-application FB shown in Figure 17. The composition of the sub-application later shown and explain in Figure 18 had been designed taking into account two-way communication. The sub-application FB can be included between any two communication FB's communicating with one another, and will ensure communication reliability between the two FB's. The sub-application feature of the IEC 61499 standard was used because of it's capability to permit distributed deployment of the FB's composing the sub-application.

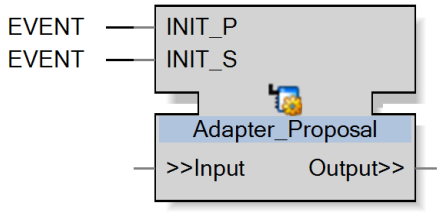

Figure 17: Sub-Application Function Block Interface.

As shown in Figure 17, the sub-application FB has been equipped with a single input and output adapter connection which has been incorporated to reduce the complexity at the application layer. Adapter interfaces will be based on the data flowing between the two communicating FB's. Along with the adapter input and output the block also contains 2 initialisation inputs. Separate initialisation inputs we added because of the developed blocked will be handling communication between two different devices, hence both the devices would individually initialisation the respective side of the handshake mechanism.

Shown in Figure 18 is the internal composition for the developed sub-application FB. To demonstrate the easy plug-and-play functionality of the developed mechanism, the sub-application FB was inserted in the existing control application for the EnAS demonstrator. As demonstrated in Figure 18, the developed FB has been introduced between the existing conveyor blocks 3 and 4 from the EnAS control application which was explained above in Figure 7.

The existing adapter interface between the conveyor blocks has been used as the interface for the sub-application which helps with the smooth addition of the reliability mechanism. Further more; in the figure, the deployment of the internal blocks has been marked. The 'Handshake_Plug Composite FB' inside the composition, performs the role of the sender for the messages going from the conveyor 3 block to the conveyor 4 block, hence it has been mapped to the conveyor 3 Ice-Block. Whereas the 'Handshake_Socket composite FB' is mapped to controller 4 ice-block because it acts as the sender for the messages conveyor 4 block wants to send to conveyor 3.

Figure 19, an extension of Figure 18, presents the internal structures for the Handshake_Plug and Handshake_Socket composite FB's. Each of the blocks, Handshake_Plug and Handshake_Socket composite FB's, are equipped with both the

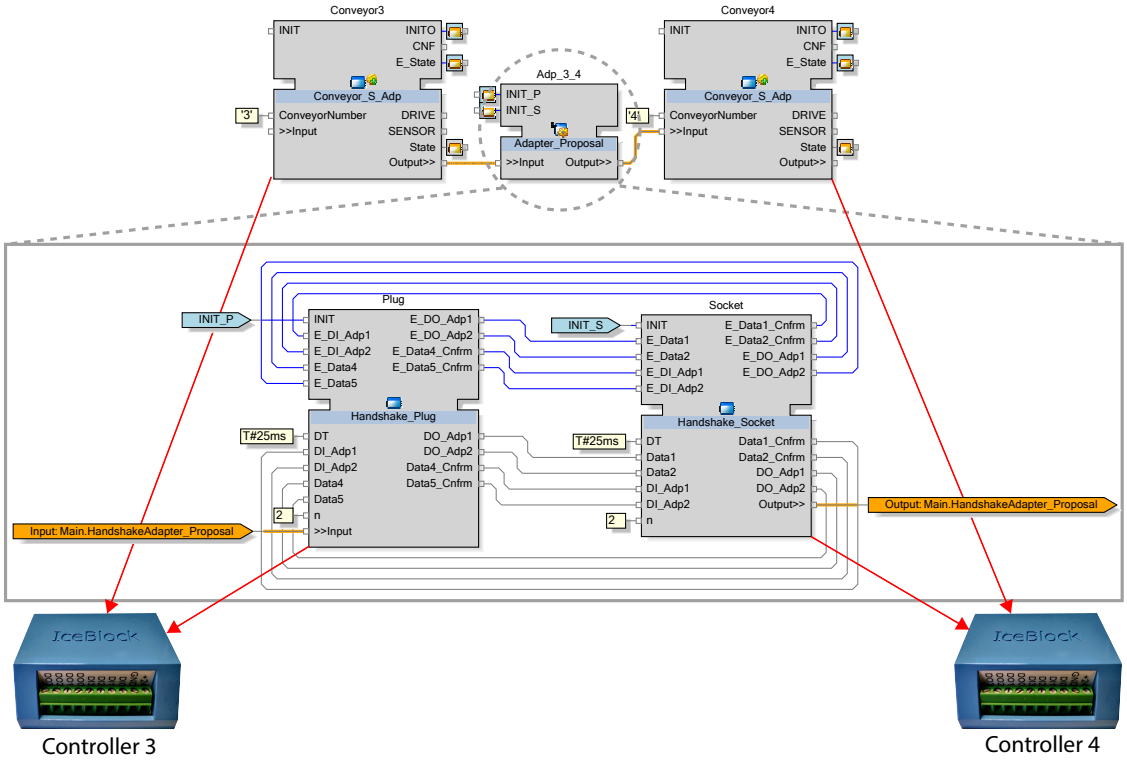

Figure 18: Sub-Application FB internal composition and device deployment.

handshake sender and the receiver FB's developed for the verification system. The composition of both the composite FB's has been designed keeping in mind asymmetrical composition. These blocks, as will be further explained using, use-cases in section 6.1 and 6.2 are capable of handling different messages in different formats being communicated in either directions at the same time.

To demonstrate the working of the asymmetrical communication via the composite blocks, communication and verification of messages moving from conveyor 3 to 4 will be demonstrated with the help of `STRING Control` commands in section 6.1, whereas messages coming from conveyor 4 to conveyor 3 have been exemplified with the help of `BOOLEAN` sensor values in section 6.2.

6.1 Communication from Conveyor 3 to Conveyor 4

`STRING Control` commands from Conveyor 3 FB are received using the adapter link at `E_Data1` & `Data1` input event and variable respectively. The received command is further passed on to the 'Sender_String' FB which has been designed to perform the handshake message verification on `STRING` type data. The processed message, which is the `STRING Command` and the message-ID, are sent across to the receiver using the outputs E_DO_Adp1 and DO_Adp1 as shown in Figure 19.

The E_DO_Adp1 event output and DO_Adp1 output variable are connected to `E_Data1` & `Data1` of the 'Receiver_String' FB present in the socket composite FB, which performs the handshake mechanisms receiver operation and forwards the

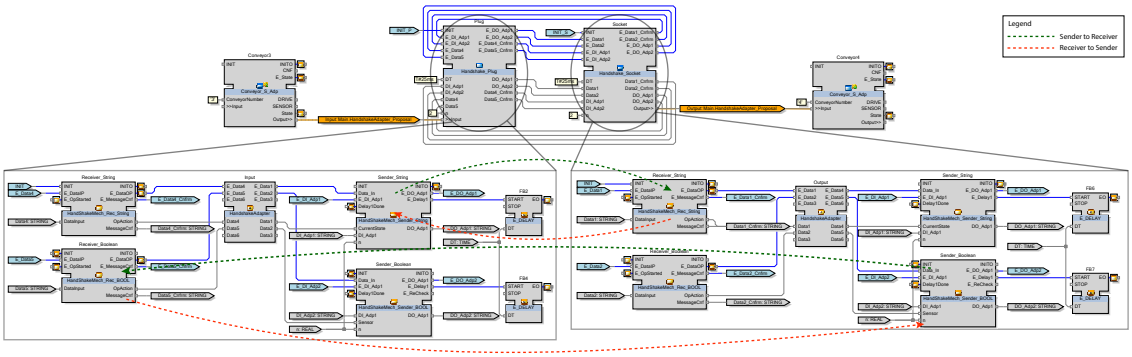

Figure 19: Composition of the Handshake_Plug and Handshake_Socket Composite Function Block.

`STRING Control` command downstream to conveyor 4 and sends back a confirmation to the 'Sender_String' FB via the E_Data1_Cnfrm and Data1_Cnfrm outputs.

Since the 'Sender_String FB' in mapped to the controller 3 and the 'Receiver_String' FB is mapped to the controller 4 because of the presence of the sub-application technology, the proposed handshake mechanism works across distributed devices and in the case of loss of communication in either of the direction, the mechanism will act according to the failure and ensure accuracy in the system.

6.2 Communication from Conveyor 4 to Conveyor 3

To ensure smooth operation of delivery services, `BOOLEAN` sensor values need to be communicated from one conveyor agent to another. In this case, we demonstrate communication of the conveyor 4 sensor values to the conveyor 3 agent. `BOOLEAN` sensor values received via the `E_Data5` & `Data5` channel of the adapter link are processed by the 'Sender_Boolean' FB present inside the Socket composite FB of the sub-application. The 'Sender_Boolean' FB has been designed to perform handshaking operations on the `BOOLEAN` data type.

The processed sensor values along with the message-ID are then transmitted to the conveyor 3 agent through the handshake mechanisms receiver FB which is the 'Receive_Boolean' FB present inside the plug composite FB. This receiver specifically designed to decode messages for `BOOLEAN` data types, isolates the sensor value from the message-ID, and passes the sensor value to the conveyor 3 agent via the adapter link.

Based on the cases presented above and the communications highlighted using the green and red dotted lines in Figure 19, the idea behind the sub-application and it's operation has been demonstrated. This developed sub-application FB used to ensure reliability and accuracy of communication was introduced between all devices in the EnAS control application and the updated application can be seen in Figure 20.

This concept and idea can be applied for N number of signals being transported in either of the directions. For each signal the system would communicate, a pair of

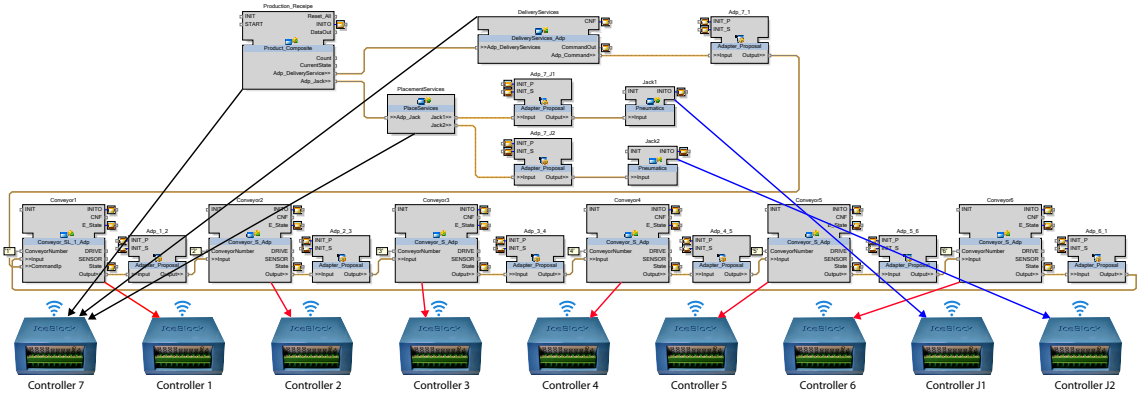

Figure 20: EnAS control application with Sub-Application.

sender and receiver handshake verification FB's would be required on each side of the adapter i.e. plug and socket. For example we need to communicate N commands from controller 3 to 4, we will need N number of handshake sender FB's on the plug side of the adapter and N number of handshake receiver FB's on the socket side of the adapter. To enable two-way communication or transportation of data the same approach will be mirrored. There will be a need of N number of handshake sender FB's at the socket of the adapter communicating with N number of handshake receiver FB's at the plug of the respective block.

7 Monitor to Check States of Agents

Monitoring of agents and their performance in larger automation systems is of extreme importance to ensure reliability and safety across the factory floors. Operators from control room should be able to monitor production processes, the devices used or the agents in command, virtually i.e. without physically going to check each component. The assumption here is that there is at least one consistency condition which has to be monitored and flagged by the monitor when violated.

In this section we explain the developed monitor to verify the operation of each agent and also ensure the operation of the handshake message verification mechanism being used between each agent. The operation of the monitor is such that, whenever the operator or the controller would want to monitor the respective sequence or part in question, they can enable the monitor using the HMI Control Panels. The designed monitor would operate on top of the existing applications communication and would not modify or change any of the information being generated by the control application. Since the monitor does not control any aspect of the control application, it hasn't been included in the SOA layered illustrations and explanations in previous sections.

The monitor can generate what is called an "Monitor Violation Event" upon detecting anomalies in the monitoring condition. The generated violation event can be used to notify the operator via the HMI panel or any other means based on the application. In the case of a monitor violation, the monitor itself will not intervene with the control of the process. Decisive action based on the monitoring results could be part of the human operators work or another part of the application.

Shown in Figure 21 is the Monitor FB developed to monitor the operation of each agent. From the "Planning Services Layer" the FB receives as input the desired state i.e. what is expected out of each agent, along with which it also receives the operational state of each agent. The monitor checks whether each agent has attained the desired state and raises a violation event and an error output only if there is detected error in the states of the agent.

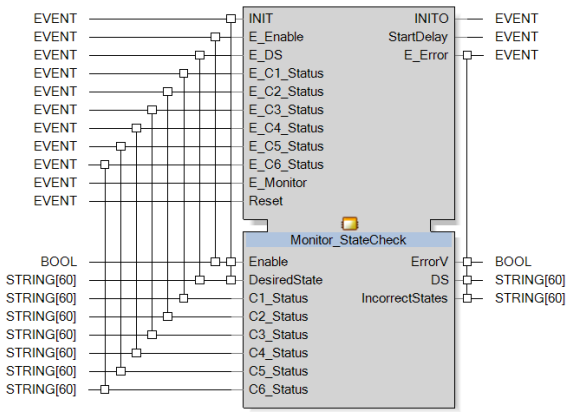

Figure 21: Interface for the Monitor.

Figure 22 shows the SM for the monitor and highlights the used algorithms. As

stated above the SM is activated via the Enable Boolean signal using the HMI which results in the SM entering the 'Wait' state where it waits for the desired command. The desired command is the command sent via the planning services layers in the application to the agents in the execution services layer, in our case, the commands are generated via the delivery services layer. The monitors receives the desired command at the same time as it is sent to the respective agents downstream.

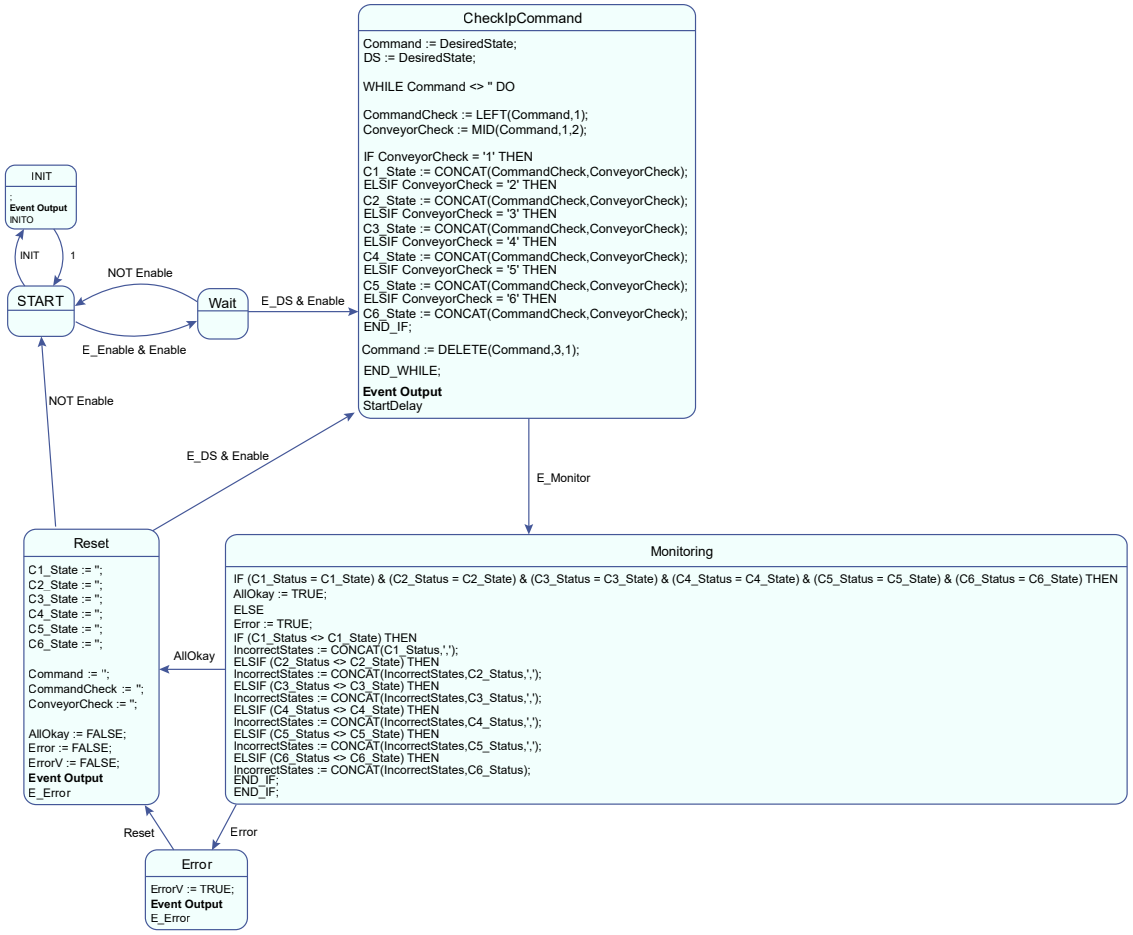

Figure 22: Monitoring State Machine.

Upon reception of the desired command, the 'CheckIpCommand' algorithm analyses the command and stores the desired state of each agent in a set of internal variables, after which it initiates a delay period before entering the monitoring state. Due to the distribution of devices and the network channel being congested, there can be numerous re-transmission attempts by the handshake message verification system to deliver the message from one agent to another, hence the delay period has been introduced, to give the agents some time to attain the desired state and then report the achieved state to the monitor. The delay time period can be pre-decided and set as constant before deployment or can be modified during runtime based on the current network performance.

Upon receiving the 'E_Monitor' event input from the Delay services, the monitor jumps to the verification state where it compares the desired states deduced in the

previous state with the reported states by each of the agents. If the verification is successful and there are no detected anomalies, the monitor resets and awaits the new commands, whereas if there is an error, the monitor jumps to a state of error and waits for the human operator to take action and reset the monitor. The need for manual reset or action in cases of error will be further explained in the consequent cases presented below.

The operational flow of the system along with the monitors has been shown in Figure 23, and the basic assumption in this figure is that the monitor is enabled. As explained in the section above all the agents and the delivery services blocks operate on different devices and communicate over the regular 2.4GHz WiFi. The monitor FB has been deployed to the same device as the HMI and delivery services(in our case controller 7) because the monitor does not control any direct hardware component and is part of the higher level layers.

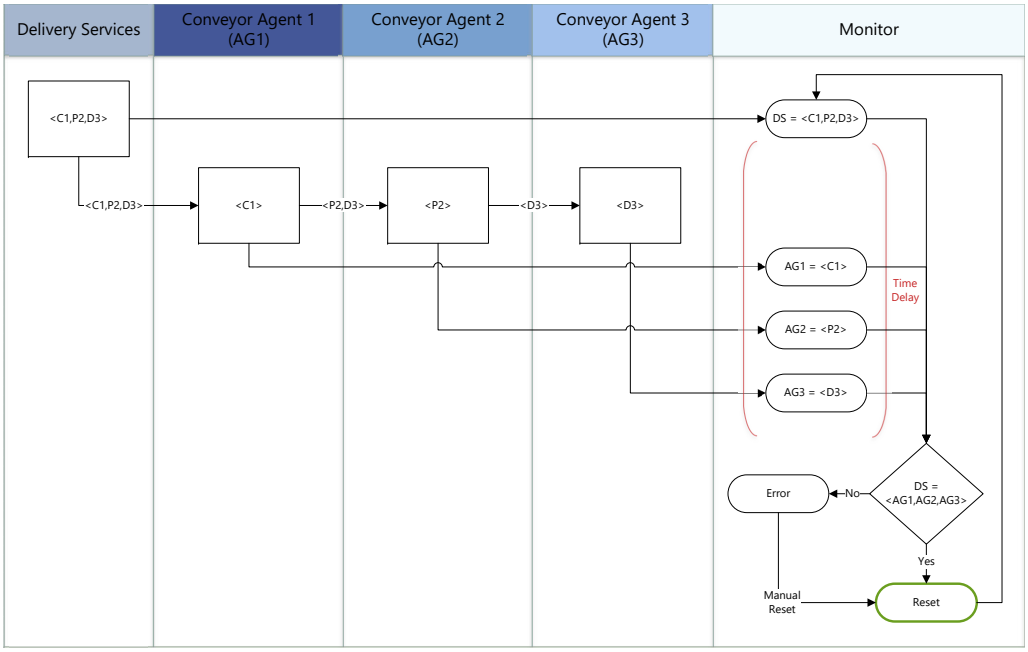

Figure 23: Monitoring Regular Operation.

The command generated by the delivery services is sent across to the monitor and to the first agent in queue. As soon as the monitor receives the command and processes it in the CheckIpCommand state, it initiates a delay period[5] before it evaluates the states received by the agents. From Figure 23, we see that all the 3 conveyor agents reported their respective states within the time delay period of the monitor, after the completion of which, the monitor compared the received desired state from the delivery services FB with the state of each agent and concluded normal operation, hence automatically jumping to the reset state and waiting for the next round of checking.

[5]In our case study : 500ms is the delay period used i.e. permitting 20 re-transmission attempts across all the agents.

This monitor not only checks the operation of each agent, but has also been used to verify the operation of the handshake mechanism. In the following subsections we discuss with demonstrated scenarios operation of the monitor in the two cases and also highlight the need for human intervention in case of errors.

7.1 Detecting Agent Failure

To demonstrate this case, we assume that conveyor agent 2 has failed i.e. the device could be spoilt, or the device could have lost connection to the WiFi and cannot communication with other agents connected upstream or downstream which has been represented in the swim-lane diagram in Figure 24. As demonstrated the processed command from Agent 1 is not delivered to Agent 2 and thereby the command also does not reach the conveyor 3 i.e. agent 3. In this case, since agent 2 was not operational it will not respond it's state to the monitor and the agent 3 reports a blank state.

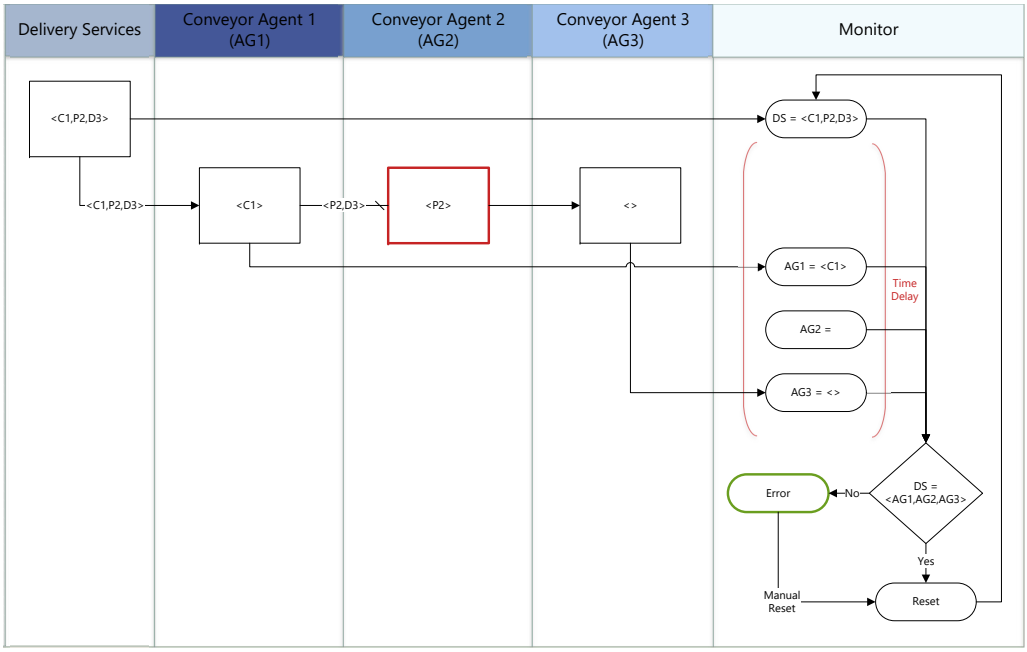

Figure 24: Agent failure condition.

After the time delay period elapses, the monitor compared the reported states and desired state, and jumps to the error state indicating a failure with the agents. Upon entering the error state, the FB highlights the error conditions to the controller i.e. the states which were not report, in this case <P2,D3>. The monitor waits for a manual reset by the user before beginning the monitoring again.

7.2 Verification of the handshake mechanism's operation

Since the agents and the monitor operate on different devices, the monitor has been used as a testing bed to further verify the working of the handshake message

verification system. The event and data connections between the monitor and the agents does not pass through the handshake message verification system i.e. the monitors and agents were directly connected with one another.

To demonstrate this we assume communication through agent 2 is hampered i.e. we assume the channel could be overloaded and there can be packet or information loss. Figure 25 showcases the above mentioned situation in which the communication from agent 2 is hampered. As demonstrated information from agent 2 is passed on to agent 3 but agent 3 reports it with a certain delay as compared to regular case from Figure 23 and in the same we also see that the state report message from agent 2 to the monitor does not go through and the monitor does not receive anything.

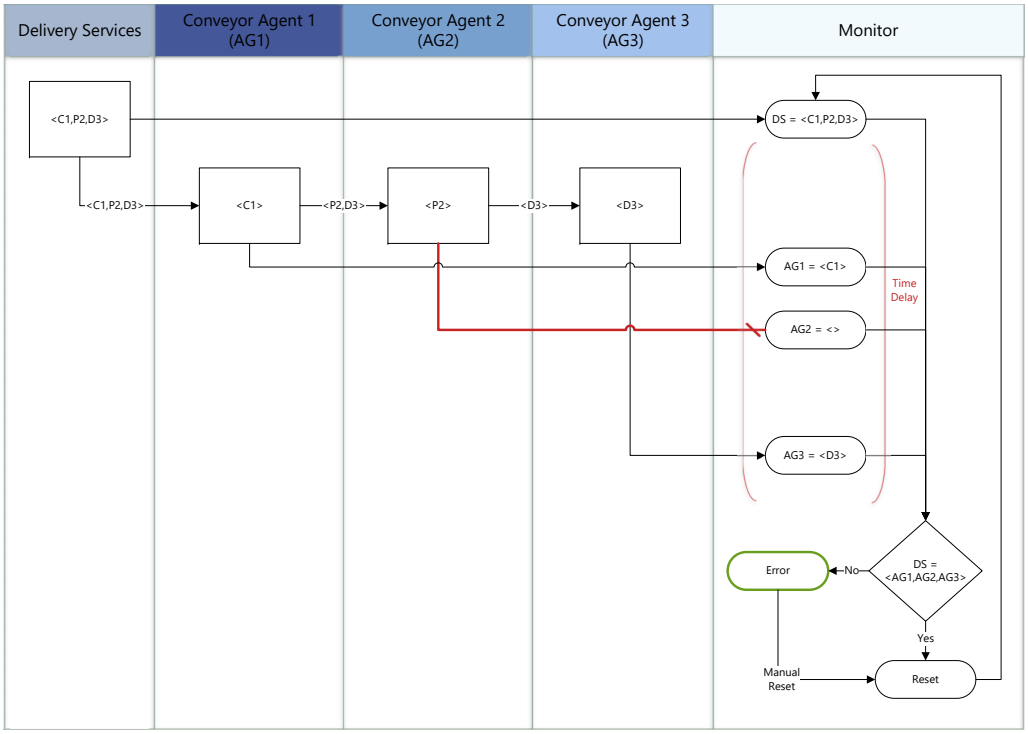

Figure 25: Monitor: Handshake Verification.

Upon monitoring, the monitor would throw an error because of lack of state information from agent 2. Based on the design of the monitor, this error would have to be verified by a human operator, which upon analysing the system would conclude that the application proceeded as normal, but instead this was an communications error between agent 2 and the monitor. Using this information, appropriate actions such as reduction of communications load or reduction of data quality can be taken to ensure reliability across agent 2.

8 Testing Scenario

To test the developed handshake message verification system and the monitors to ensure reliability, the EnAS demonstrator was run for continuous production rounds where red and green coloured work-pieces were produced to demonstrate the production in the industry and test the developments. The number of rounds the demonstrator completed before giving an error was recorded and analysed.

Figure 26 shows the sequential task performed, in which upon starting, the empty production cup is carried from conveyor 3 to conveyor 5 (C3_to_C5). Upon reaching the conveyor 5 sensor which is position in front of the pneumatic jack 2, the jack places the green work-piece on the empty production cup. Once the green work-piece has been placed, the production recipe block in the top-most layer of the SOA which is production services layer, commands the conveyors to carry the production cup with the green work-piece from conveyor 5 to conveyor 1, followed by conveyor 1 back to conveyor 5.

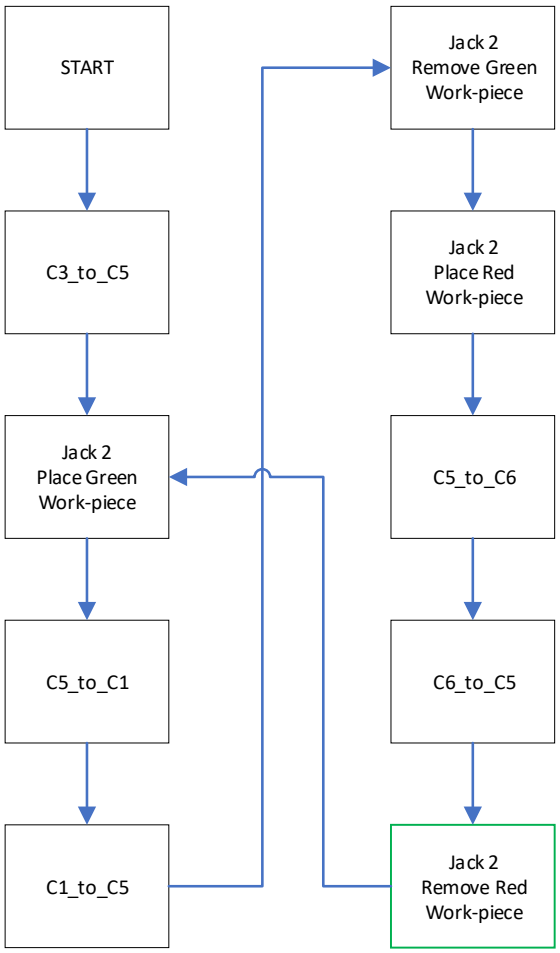

Figure 26: Production sequence flow chart.

When the work-piece has again reached the jack 2 production station at conveyor 5, it is first instructed to remove the existing green work-piece and place it back on

the sledge, followed by picking the red work-piece from the sledge and placing it on the production cup. The production cup with the red work-piece is then carried to conveyor 6, followed by the next set of operation which is taking the production cup from conveyor 6 back to conveyor 5. At the production island, jack 2 then removes the red work-piece and places in on the sledge. This operation marks 1 complete production round, following which the process is looped back to placing the green work-piece as can be seen in Figure 26.

As can be seen above in the flowchart and description, the delivery services, which are responsible of carrying the production cup and work-pieces around the EnAS have been accomplished using different delivery routines by breaking deliveries into various segments. Table 1, contains the delivery sequences used in the production scenario. Each delivery sequence is passed on the from the production recipe block to the delivery services block in the planning services layers.

Table 1: EnAS Delivery Sequences and commands

Delivery Sequence	Delivery Command
C3_to_C5	C3,P4,D5
C5_to_C1	C5,P6,D1
C1_to_C5	C1,P2,P3,P4,D5
C5_to_C6	C5,D6
C6_to_C5	C6,P1,P2,P3,P4,D5

The delivery sequences are associated with a delivery command which has been used between the layer 2 which is the planning services and layer 3 which is the execution services. These sequences and commands of varying lengths were introduced to test the robustness of the handshake message verification system and observe if changing length of commands being processed via the handshake mechanism sender and receiver block affected the operation of the reliability mechanism.

9 Results and Conclusions

9.1 Handshake Message Verification

The production scenario highlighted above in section 8 was run for continuous rounds and the number of successfully completed rounds and number of system failures i.e. messages not passing through due to error in communication was recorded. The testing of the handshake mechanism was performed in 3 different sets based on the development of the various handshake sender SM's.

First round of testing involved deploying the handshake sender SM explained in section 5.5.1, in which the sender would re-check for updated values at each re-transmission. Results for the testing have been highlighted in table 2, where we calculated that the EnAS demonstrator ran for 37 rounds before giving an error and halting the production scenario. The analysis of the FB network upon error showcased the need for re-transmission of old messages, because in cases when the old message was not re-transmitted and the new message was received and instantly sent, the hardware components did not receive the old message and the desired operation was not performed, hence the system would either stop in this case or a wrong product was produced.

Table 2: Testing Results

	N	No. of Rounds	Errors	No. of Rounds/Error
Case 1	-	259	7	37
Case 2	2	365	9	41
Case 2	3	682	12	57
Case 2	5	386	9	43
Case 2	10	289	11	26
Case 3	3	1365	2	685

Based on the analysis the need for a certain number of re-transmission of old messages was highlighted and the SM developed and explained in section 5.5.2 for the same, was put to test. The handshake sender mechanism would re-transmit the old message 'N' number of times before considering a new message. To develop the most efficient scenario for the EnAS demonstrator and the application, the optimal number of permitted re-transmissions had to be tested and therefore the SM shown in Figure 12 was tested numerous times by changing the value of N each time. Optimal number of re-transmissions was need to ensure maximum reliability at the cost of the least possible delay in the production, because each re-transmission in our testing would introduce a 25ms delay period in the production scenario.

Showcased in Figure 27, is the graph plot for the number of production rounds completed before giving one error for the various number of re-transmissions tried. The table 2 also highlights the results for the testing of the case, where based on the results achieved we can see the a minimum of 3 re-transmissions of the old-message was needed to ensure maximum reliability in the production scenario.

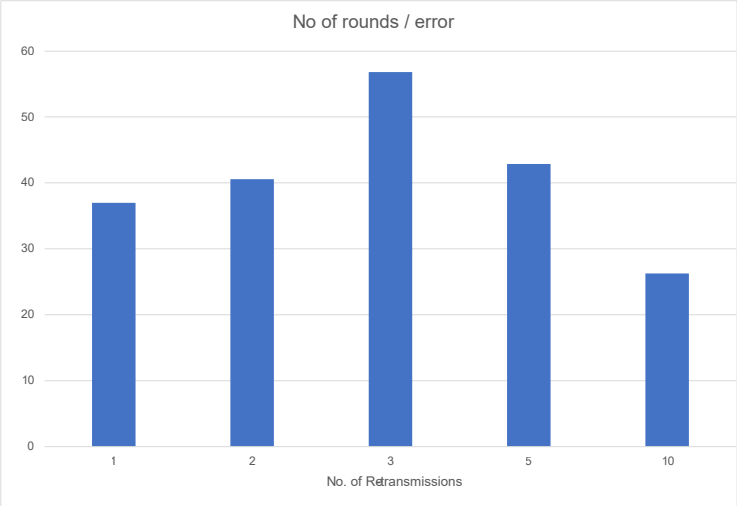

Figure 27: Graphical Representation of Number of rounds completed.

In the case when the number of re-transmissions permitted was set at 10, which means after re-transmitting the old-message 10 times, the SM would check for an updated command. For example, if a new command was received during the 4th transmission of the old command, and the receiver sends back a confirmation for the 6th transmitted command, the re-transmission would be halted and then the sender SM would jump to the 'WAIT' state and would wait for an updated command based on the SM in Figure 12. Since in this case, the number of re-transmissions was 10, the handshake mechanism sender block would have checked for an updated command at the 10th transmission, but because re-transmission halted at the 6th transmission, the SM did not get into the state 'CheckCommand' to re-check the updated command, hence missing the already received new command and generating an error. Whereas, in the case when N was 3, the updated command was more frequently checked because at every 3rd re-transmission the SM would check for an updated command, which thereby resulted in the program completing more successful production rounds with the least amount of errors.

According to the argument above, in the case of number of re-transmissions being set to 2 should have resulted in an even higher accuracy because the SM would check the commands even more frequently, but as we can see in Figure 27 the number of successful rounds has drastically dropped in this case. This intern concluded that the minimum number of re-transmissions of old-messages needed for the system to operate properly is 3.

Based on further analysis of the testing done for case 1 and 2, a new SM discussed in section 5.5.3 and showcased in Figure 13 was developed, which would combine the idea's provided in both SM's shown in Figure 11 and 12, and also resolve the error of not sending the new message in the case of received confirmation within the permitted number of re-transmissions. As highlighted in Table 2, the results for case 3 were exceptional improved in which the demonstrator ran 685 rounds before giving an error.

9.2 Sub-application Prototyping

The developed prototype using the sub-application FB showcased the ease of system assembly and integration. The developed handshake mechanism was easily integrated into the existing control application for the EnAS demonstrator. On comparing Figures 7 and 20, we can see that the developed block was easily integrated between any two FB's mapped to different devices and the internal compositions of the sub.application FB were mapped accordingly as well, which in-turn ensured the reliability across the devices communicating over a wireless channel.

The sub-application FB ensured the handshake mechanism and the FB's from the SOA control application were separate from one another, which enabled quick debugging of either of the algorithms, reliability or control. The developed sub-application FB contained a mature re-transmission algorithm, thereby assuring the engineers and operators on the factory floor that any fault in the system would not be because of the additional reliability mechanism, rather could be a fault at the control algorithm layer or hardware layer.

The future aspect of this work is to include the developed reliability mechanisms within the existing adapter connections and not have to use additional FB's to provide the additional functionality.

9.3 Verification of reliability using Monitors

To verify the operation of the handshake mechanism, an additional monitor explained in section 7 was developed. The additional monitor was also deployed to controller 7 which also houses the production recipe FB and the planning services blocks. Even though, the conveyor agents deployed to their respective controllers sent information to the monitor, they were not passed through the developed sub-application FB, rather they directly communicated their values to the monitor.

While the continuous testing of the production scenario, the operation of the monitor was visualised on the HMI, where-in the monitor gave errors numerous times. The error given by the monitor signified that one or more of the conveyor agents did not attain their desired states, according to which the production should have halted on the particular conveyor agent.

On cross-referencing the error with the actual EnAS demonstrator, it was observed that the production still continued, and the agents attained their desired state. Upon, further analysis of the FB network, it was revealed that the monitor gave an error because it would check after a 500ms delay period, which means, allowing the agents 500ms to attain their desired states but the production still continued because the agents did achieve their desired state, only with higher number of re-transmissions, thereby taking more than 500ms to attain the state. This signified the importance of the developed handshake message verification system.

10 Summary

In this thesis, various aspects of reliability were examined in the literature review and based on the analysis of those, to improve the communication between two or more controllers communicating over a standard lossy wireless channel, a handshake message verification system based on re-transmissions was developed.

The handshake message verification system ensured reliability of communication by re-transmitting old messages which were lost during transmission. The development of this handshake mechanism was done based on the idea of modularity and re-use of functions to improve reliability in software. Also, the developments which results in successful and reliable communication were done keeping in the mind the existing lossy channel. To have easy of system assembly and on factor floor integration, the developments were done at the application layer and did not require any hardware modification at the network layer or controller level.

This work [22] has been submitted to the IEEE Access Journal.

References

[1] "IEC 61499: Function blocks—part 1 architecture," standard, Int. Electrotech. Commission, Geneva, Switzerland, 2012.

[2] "Standard IEC 61131-3: Programmable controller - part 3: Programming languages," standard, Int. Electrotech. Commission, Geneva, Switzerland, 1993.

[3] S. Patil, D. Drozdov, and V. Vyatkin, "Adapting software design patterns to develop reusable IEC 61499 function block applications," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pp. 725–732, IEEE, 2018.

[4] P. Jhunjhunwala, U. D. Atmojo, and V. Vyatkin, "Towards implementation of interoperable smart sensor services in IEC 61499 for process automation," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 1409–1412, IEEE, 2020.

[5] P. O'Connor and A. Kleyner, *Practical reliability engineering*. John Wiley & Sons, 2012.

[6] "IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems - part 1: General requirements," standard, Int. Electrotech. Commission, 2010.

[7] C. S. Wasson, *System engineering analysis, design, and development: Concepts, principles, and practices*. John Wiley & Sons, 2015.

[8] H. Pham, *System software reliability*. Springer Science & Business Media, 2007.

[9] "ISO/IEC 90003:2014 software engineering — guidelines for the application of ISO 9001:2008," standard, Int. Electrotech. Commission, 2014.

[10] J. M. Spinelli *et al.*, "Reliable communication on data links," 1988.

[11] M. A. Mahmood, W. K. Seah, and I. Welch, "Reliability in wireless sensor networks: A survey and challenges ahead," *Computer networks*, vol. 79, pp. 166–187, 2015.

[12] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pp. 78–89, 2004.

[13] N. Tezcan and W. Wang, "Art: an asymmetric and reliable transport mechanism for wireless sensor networks," *International Journal of Sensor Networks*, vol. 2, no. 3-4, pp. 188–200, 2007.

[14] G. Frey and T. Hussain, "Modeling techniques for distributed control systems based on the IEC 61499 standard-current approaches and open problems," in *2006 8th International Workshop on Discrete Event Systems*, pp. 176–181, IEEE, 2006.

[15] G. Cengic, O. Ljungkrantz, and K. Akesson, "Formal modeling of function block applications running in IEC 61499 execution runtime," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, pp. 1269–1276, IEEE, 2006.

[16] H.-C. Lapp, C. Gerber, and H.-M. Hanisch, "Improving verification and reliability of distributed control systems design according to IEC 61499," in *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pp. 1–8, 2010.

[17] M. V. García, E. Irisarri, F. Pérez, E. Estévez, and M. Marcos, "An open cpps automation architecture based on IEC 61499 over opc-ua for flexible manufacturing in oil&gas industry," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1231–1238, 2017.

[18] Z. E. Bhatti, P. S. Roop, and R. Sinha, "Unified functional safety assessment of industrial automation systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 17–26, 2017.

[19] U. D. Atmojo, V. Vyatkin, and Z. Salcic, "On achieving reliable communication in IEC 61499," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 147–154, 2018.

[20] F. Weehuizen and A. Zoitl, "Using the cip protocol with IEC 61499 communication function blocks," in *2007 5th IEEE International Conference on Industrial Informatics*, vol. 1, pp. 261–265, 2007.

[21] A. Zoitl and R. Lewis, *Modelling control systems using IEC 61499*, vol. 95. IET, 2014.

[22] P. Jhunjhunwala and V. Vyatkin, "Proposing and prototyping an extension to the adapter concept in the iec61499 standard(status = submitted)," *IEEE-Access*, 2021.